PREPARED FOR THE
DEPARTMENT OF THE ARMY
Contract DAHC19-69-C-0017

RAC-TP-445
FINAL DRAFT
OF VOL II

JANUARY 1972

A Methodology for Optimal Planning over Time

Volume II

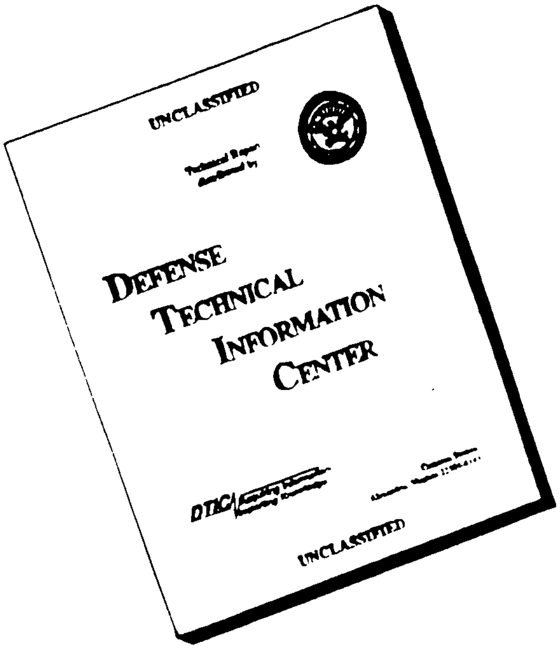
Appendices A, B, C, D, and E in Support of Volume I

by Charles A. Allen
Beverly D. Causey
James E. Falk
Ronald G. Magee
Charles W. Mylander
Ronald New, *Project Director*
John D. Pearson
Philip D. Robers

FINAL DRAFT

Copy__________of 165

RAC Research Analysis Corporation


UNCLASSIFIED
DEFENSE
TECHNICAL
INFORMATION
CENTER
DTIC
UNCLASSIFIED

DEPARTMENT OF THE ARMY
OFFICE OF THE CHIEF OF RESEARCH AND DEVELOPMENT
WASHINGTON, D.C. 20310

DARD-ARS

1. Volume I, "A Methodology for Optimal Planning Over Time" and Volume II, "Appendices A, B, C, D and E in Support of a Methodology for Optimal Planning Over Time - Volume I," were prepared by the Research Analysis Corporation for the Combat Systems Group, United States Army Combat Developments Command, and document RAC study 011.310, "Aircraft Systems Least Cost Phase-In." Copies of these reports are forwarded for your retention and use.

2. The methodology described in these volumes was developed to meet in part the need of the US Army to determine an optimal plan for phasing in new aircraft systems to meet its worldwide commitments yet remain within budgetary constraints. It provides to planners a tool for use in planning situations involving consideration of large numbers of alternative systems and combinations of tasks.

3. The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

FOR THE CHIEF OF RESEARCH AND DEVELOPMENT:

Stanley R Meeken
STANLEY R. MEEKEN
Colonel, GS
Chief, Studies and Analyses Division

Published January 1972
by
RESEARCH ANALYSIS CORPORATION
McLean, Virginia 22101

CONTENTS

VOLUME I

SUMMARY S-1

INTRODUCTION I-1

CHAPTER 1 - PROBLEM STRUCTURE 1-1

CHAPTER 2 - A QUALITATIVE DESCRIPTION OF BRANCH AND BOUND 2-1

CHAPTER 3 - MATHEMATICAL DESCRIPTION OF PROBLEM AND ITS SOLUTION 3-1

THE CONSTRAINT SET 3-1

Materiel Balance Constraints - Consistency Constraints- Vintage Constraints - Master Variable Relation Constraints - Cost Constraints

THE OBJECTIVE FUNCTION 3-6

Primary Cost Categories - Secondary Cost Categories

THE SOLUTION PROCEDURE 3-10

Linear Envelopes - Bounding the Solution - Partitioning into Subsets - The Algorithm

CHAPTER 4 - AN OPERATIONAL DESCRIPTION OF THE BRANCH AND BOUND SOLUTION PROCEDURE 4-1

PROGRAM INTERFACING 4-1

THE MATRIX GENERATOR 4-5

Program Logic - Core Allocation - User's Subroutine (YRCCST) - Input Formats - Output Description

THE MAIN PROGRAM 4-21

Program Logic - Core Allocation - User's Subroutine (GETPHI) - Input Formats - Output Description

THE REPORT GENERATOR 4-34

Program Logic - Core Allocation - User's Subroutine (YRCOST) - Input Formats - Output Description

REFERENCES R-1

FIGURES: VOLUME I

1-1 A Typical List of Alternative Vehicle Arrays Which Could Service a Mission Group 1-1

1-2 Eight Independent Groups Each with Twenty Equally-Effective Alternatives 1-3

1-3 The Mission Group Tableau 1-5

FIGURES: VOLUME I (continued)

2-1 A Concave Cost Function 2-1

2-2 Partitioning the Total Solution Set Into Subsets 2-3

3-1 Cost Function 3-11

3-2 A Linear Envelope of a Concave Function with Discontinuity at Origin 3-13

3-3 The Algorithm 3-17

4-1a Control Cards to Execute All Programs as a Single Job 4-3

4-1b Control Cards to Execute Programs as Separate Jobs 4-4

4-2 System Macro Flowchart 4-6

4-3 Symbolic Naming Convention 4-7

4-4 Positive Integer Code 4-8

4-5 Basic Cards 4-13

4-6 Vehicle Table 4-14

4-7 Period Table 4-15

4-8 Task Table 4-16

4-9 Deck Structure for Matrix Generator 4-20

4-10 Input Data Formats 4-27

4-11 Deck Structure for Main Program 4-30

4-12 Deck Structure for Report Generator 4-37

CONTENTS
VOLUME II

APPENDIX A - SAMPLE PROBLEM A-1

APPENDIX B - SUBROUTINE DESCRIPTIONS B-1

APPENDIX C - FLOWCHARTS C-1

APPENDIX D - PROGRAM LISTING D-1

APPENDIX E - ERROR MESSAGES E-1

GLOSSARY G-1

REFERENCES R-1

FIGURES: VOLUME II

A-1 Alternative Fleet Mixes for 1972 A-1

A-2 Capital Equipment (Truck) Requirements as a Function of Time A-2

A-3 The Simplified Objective (cost) Function for the Sample Truck Problem A-4

A-4 The Sample Problem Data Deck for the GENLCP Program A-8

A-5 GENLCP Output For Sample Problem, Parts (a) - (f) A-10

A-6 The Sample Problem Data Deck For the BECAV2 Program A-17

A-7 A Branching Tree For the Sample Problem A-19

A-8 The Sample Problem Data Deck For the REPGEN Program A-20

A-9 REPGEN Output For Sample Problem, Parts (a) and (b) A-21

A-10 A Sample Optimal Plan For the Ace Trucking Company A-24

APPENDIX A

A SAMPLE PROBLEM

We have chosen a relatively small sample problem to illustrate an implementation of the branch-and-bound algorithm and its corresponding program. We will structure the problem from a user's viewpoint, formulate the objective function and the constraint set, illustrate the preparation of the data decks and user's subroutines, and explain the logic of the results. The problem to be described here has been discussed in a previous document.[4]

The Ace Trucking Co., in planning for next year's workload, estimates that the company can serve its customers with a fleet of 67 light trucks and 16 cross-country trailers. An alternative fleet was also considered consisting of 43 light and 20 medium sized trucks, together with only 7 cross-country trailers. Finally, the only other practical alternative considered was a mix of 42 medium trucks and 12 of the big trailers. The medium trucks, however, are of a new design and will not be available for next year unless the company is willing to pay a substantial premium. At first it appeared that choosing one of these three alternative fleets (shown in table form below) was

	light trucks	medium trucks	trailers
Alternative #1	67	—	16
Alternative #2	43	20	7
Alternative #3	—	42	12

Figure A-1 ALTERNATIVE FLEET MIXES FOR 1972 the only decision issue. However, it soon became clear to the planning group at Ace Trucking Co. that the investment decision should also depend on the utilization of the trucks in subsequent years, in addition to that utilization planned for the next year. And furthermore, the existing fleet of trucks was far from obsolete, even though maintenance costs on some of the older vehicles were beginning to climb. Realizing these factors, the planning group estimated the workload for their trucks over the next three years (beyond which they could not be confident of their estimates), and then prepared a requirements table like that in Figure A-2 below.

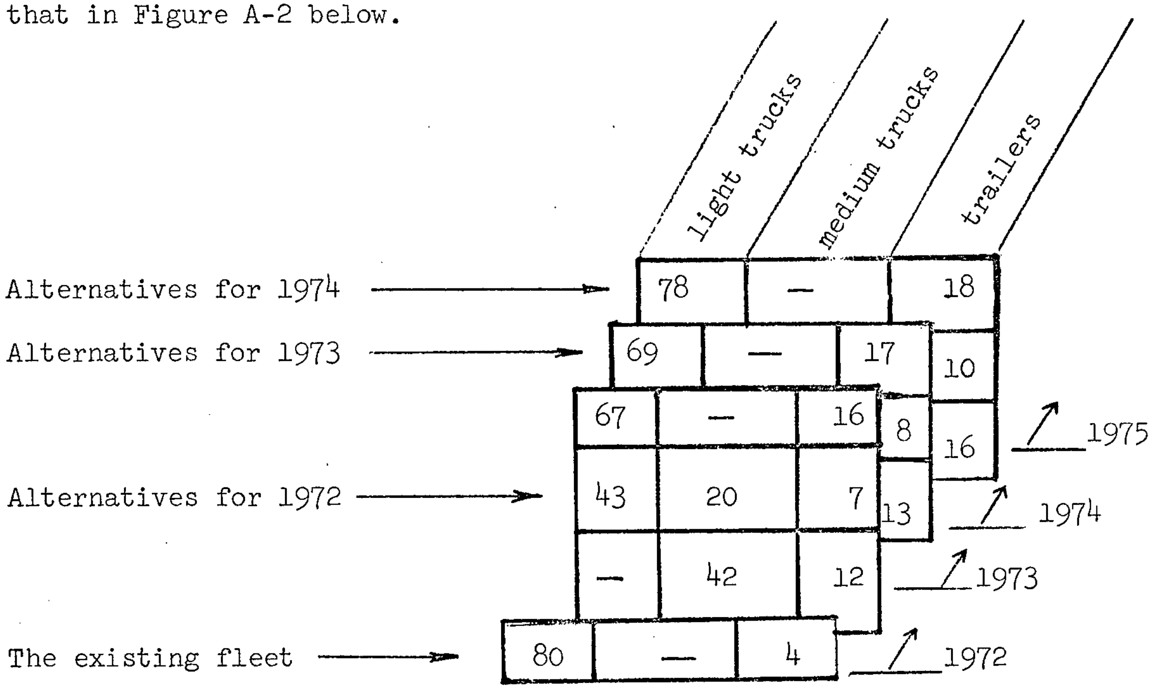


Figure A-2 CAPITAL EQUIPMENT (TRUCK) REQUIREMENTS AS A FUNCTION OF TIME

The existing fleet did not include any medium sized trucks; 60 light trucks were two years old, but the remaining 20 were purchased only last year. The inherited trailers were also two years old. Finally, based on the projected cash flow position of Ace for the next three years, it was decided that a limit (cost constraint) be placed on new truck procurements for each of the three years.* Decision time for Ace

* Costs constraints were not imposed on this problem in the previous referenced description;[6] hence, a slightly different solution was obtained.

Trucking Co. is January 1972, and the question is: what is the optimal plan over the next three-year period?

Initially, one should ask: how many alternative plans exist? If we permute three alternatives per year with three years, we develop 27 alternative plans — but this number ignores all permutations of the existing fleet as well as all concern for the "welfare" of the inherited fleet from year to year. In addition, there are questions like "buy" or "lease," ... "sell," "salvage," or "store," etc., making the number of alternative plans very large indeed (literally thousands even in this trivial problem). In fact, for real-world problems, it would not be atypical to have millions of possible alternative plans confronting the decision-maker, each involving a maze of cost factors — making evaluation of each and every plan rather impractical. Problems like this can be solved efficiently, without evaluating each and every alternative, using the mathematical programming techniques embodied within the Falk-Soland Algorithm.[2]

The objective function for this problem is developed from the general form of Fig. 3-1 (page 3-11) plus projected estimates of the cost coefficients for each cost catagory. We have only three vehicles under consideration but we artifically introduce a fourth vehicle (and call it MEDIUM*) to account for the premium charge <u>if</u> we buy MEDIUM trucks for the first year (1972). That is, the program will treat the purchase of MEDIUM trucks and MEDIUM* trucks separately, as if they were distinct. In figure A-3 we show the reduced form of the objective function. Note that,

(1) There is only one (possible) R&D charge and that is associated with the MEDIUM* vehicles (called vehicle No. 1; the LIGHT,MEDIUM, and TRAILERS are numbered 2, 3, and 4, respectively.

(2) We use <u>no</u> increase in operating cost over time (for simplicity) hence, we drop the second subscript in the c coefficients as well as the summation over k.

(3) We have introduced the specific values for the number of vehicles (N=4) and the length of the planning period (Y=3) where they appear in figure A-3.

$$\underbrace{\varphi(x) = U_1(x_1)}_{\text{R\&D Costs}} + \underbrace{\sum_{J=1}^{4} a_J x_J^{b_J}}_{\text{Procurement Costs}} + \underbrace{\sum_{J=1}^{4}\sum_{\ell=1}^{3}(-\, d_{J\ell})s_{J\ell}}_{\text{Savings due to mothballing of unneeded vehicles}}$$

$$+ \sum_{J=1}^{4}\sum_{\ell=K_J}^{0}\sum_{m=1}^{3} c_J w_{J\ell m} \qquad + \sum_{J=1}^{4}\sum_{\ell=1}^{3}\sum_{m=\ell}^{3} c_J x_{J\ell m}$$

Operating and maintenance cost

(inherited fleet) (purchased fleet)

$$+ \sum_{J=1}^{4}\sum_{\ell=K_J}^{0}\sum_{m=0}^{2}(-e_{J,(m-\ell+1)})w_{J\ell m} + \sum_{J=1}^{4}\sum_{\ell=1}^{3}\sum_{m=\ell}^{3}(-\, e_{J,(m-\ell+1)})x_{J\ell m}$$

Savings resulting from vehicle salvage

(inherited fleet) (purchased fleet)

$$+ \sum_{J=1}^{4}\sum_{\ell=K_J}^{0}(-\, f_{J,(4-\ell)})w_{J\ell 3} \qquad + \sum_{J=1}^{4}\sum_{\ell=1}^{3}(-\, f_{J,(4-\ell)})x_{J\ell 3}$$

Savings due to crediting the value of fleet owned
at the end of the planning period

(inherited fleet) (purchased fleet)

Fig A3 - The Simplified Objective (cost) Function For the Sample Truck Problem

The constraint set is developed from the general descriptions given in Chapter 3 (starting on p. 3-1).

The material balance constraints become:

$$\sum_{\ell'=K_j}^{o} \sum_{m=\ell}^{3} w_{j\ell'm} + \sum_{\ell'=1}^{\ell} \sum_{m=\ell}^{3} x_{j\ell'm} = \sum_{i\in M_\ell} \sum_{k=1}^{R_{i\ell}} u_{ijk\ell}\, p_{ik\ell} + s_{j\ell}$$

$$j = 1, 2, 3, 4$$
$$\ell = 1, 2, 3$$

Note that we have assumed no attrition ($v_{\ell-\ell'} = 1$), and have assigned the $t_{i\ell}$ factor = 1 for all mission groups.

The consistency constraints reduce to:

$$\sum_{k=1}^{R_{i\ell}} p_{ik\ell} = 1; \quad \ell = 1, 2, 3 \text{ and } i = 1, 2, 3$$

The vintage constraints for the light trucks and trailers are:

$$W_{j\ell} = \sum_{m=o}^{9+\ell} w_{j\ell m}; \quad j = 2, 4 \text{ and } \ell = -1, 0$$

where L_j has been replaced by 10 subperiods (years) for all vehicles and K_j by -1 for both inherited vehicles.

The master variable constraints are simply:

$$x_j = \sum_{\ell=1}^{3} \sum_{m=\ell}^{3} x_{j\ell m}; \quad j = 1, 2, 3, 4$$

Finally, since we have chosen to introduce cost constraints, we have

$$H_{\ell} = \sum_{j=1}^{4} a_j^o \sum_{m=\ell}^{3} x_{j\ell m} + P_{\ell} - P_{\ell-1}$$

We (i.e., Ace Trucking Co.) will set the cost constraint, H_{ℓ} = $150,000., $250,000., and $300,000., for the three years of the planning period, respectively. The linear approximation a_j^o, to the procurement cost function will be selected after inspection of the actual cost vs. quantity curve.

A careful inspection of the total constraint set for this problem will indicate a total of 24 constraints (recall that there is no materiel balance constraint for [ℓ=1, j=3] and no vintage constraint for [j=4, ℓ=0]). Similiarly, a count of the number of variables (being careful to delete those which must equal zero because of specific exclusions in this sample problem) will indicate a total of 60. The reader will note that the GENLCP Program automatically computes and prints these totals for use in the BBCAV 2 Program.

We are now ready to prepare the data decks and user subroutines. The user subroutine GETPHI is shown in the program listing on page D-32 . It is here that we describe the R&D and procurement equations for the four vehicles in the sample problem. Note that vehicle No. 1 (MEDIUM*) has an R&D (premium) charge of $300,000.* — the other three vehicles have no R&D charge and their procurement costs are simply described by concave functions of the form ax^b. Of course, other forms of concave functions could have been used.

* We have chosen to scale all costs by 10^6. This means that all final cost data should be multiplied by 10^6.

The only other user subroutine YRCOST (see page D-19) is prepared for use in the GENLCP program, then duplicated for use in REPGEN. As described previously on p. 4-10 of chapter 4, YRCOST is used to calculate the operating, mothballing, salvage, and truncation cost coefficients, c_{jk}, $d_{j\ell}$, e_j, $(m-\ell+1)$ and f_j, $(Y-\ell+1)$ respectively. For our sample problem, we use no increase in operating cost overtime (R=0.); a mothballing savings factor of R1 = 0.9; a salvage savings factor of alpha = 0.5; and truncation savings based upon a linear decay from an estimate of the purchase cost (input through the GENLCP data deck) and an assumed 10 year lifetime for each vehicle.

The data deck for the GENLCP program can now be prepared (see figure A-4).

In entry (card image) No. 1 we give the problem title, the first and last year of the planning period, and then specify the four vehicle tables, three task tables, and five period tables — the first two of which are inherited periods. Entry 2 is the VEHICLE header card for the first vehicle. Entry 3 describes the first vehicle as LIGHT, indicates an availability date of 1970 (i.e. an inherited vehicle) and finally a ten year vehicle lifetime. Entry 4 indicates that 60 light vehicles were purchased in 1970 and 20 light vehicles were purchased in 1971. Entry 5 indicates a $3,000. purchase cost estimate for purposes of calculation of the truncation and salvage value, a ten year operating cost of $120,000., zero R&D cost for the light vehicle, zero attrition for the light vehicle, and finally an estimated linear purchase cost coefficient of $3,000., respectively. This linear purchase cost coefficient estimate was based upon a study of the corresponding non-linear procurement equation for the light vehicle. In general, one should choose the cost coefficient (slope of the straight line) such that the straight line intersects the non-linear curve at or about the estimated solution value. The consequences of a poor estimate will be described shortly. Entries 6 through 15 simply complete the vehicle tables. Entries 16 through 31 describe the period tables. The first two periods (1970 and 1971) are inherited periods. In period 1972 we indicate a cost constraint of $150,000. in entry 21. Entry 22 specifies that there exists only one task in 1972 and its scale factor is 1.0.

```
SAMPLE        1972 1974     4      3      5
VEHICLE
LIGHT     1970          10
          60           20
     .003        .12          0.0         1.0         .003
VEHICLE
MEDIUM*   1972          10
     .006        .14         .300         1.0         .006
VEHICLE
MEDIUM    1973          10
     .006        .14          0.          1.0         .006
VEHICLE
TRAILER   1970          10
          4
     .01         .16          0.          1.0         .012
PERIOD
1970 1970
PERIOD
1971 1971
PERIOD
1972 1972    .15
           1            1.0
      1   1.0
PERIOD
1973 1973    .25
           1            1.0
      2   1.0
PERIOD
1974 1974    .3
           1            1.0
      3   1.0
TASK
           1            3            3
LIGHT      MEDIUM*      TRAILER
67.0       0.0          16.0
43.0       20.0         7.0
0.0        42.0         12.0
TASK
           2            4            5
LIGHT      MEDIUM*      MEDIUM       TRAILER
69.0       0.0          0.0          17.0
45.0       22.0         0.0          8.0
0.0        45.0         0.0          13.0
45.0       0.0          22.0         8.0
0.0        0.0          45.0         13.0
TASK
           3            4            5
LIGHT      MEDIUM*      MEDIUM       TRAILER
78.0       0.0          0.0          15.0
48.0       24.0         0.0          10.0
0.0        50.0         0.0          16.0
48.0       0.0          24.0         10.0
0.00       0.0          50.0         16.0
ENDTABLE
```

Fig. A-4 The Sample Problem Data Deck for the GENLCP Program

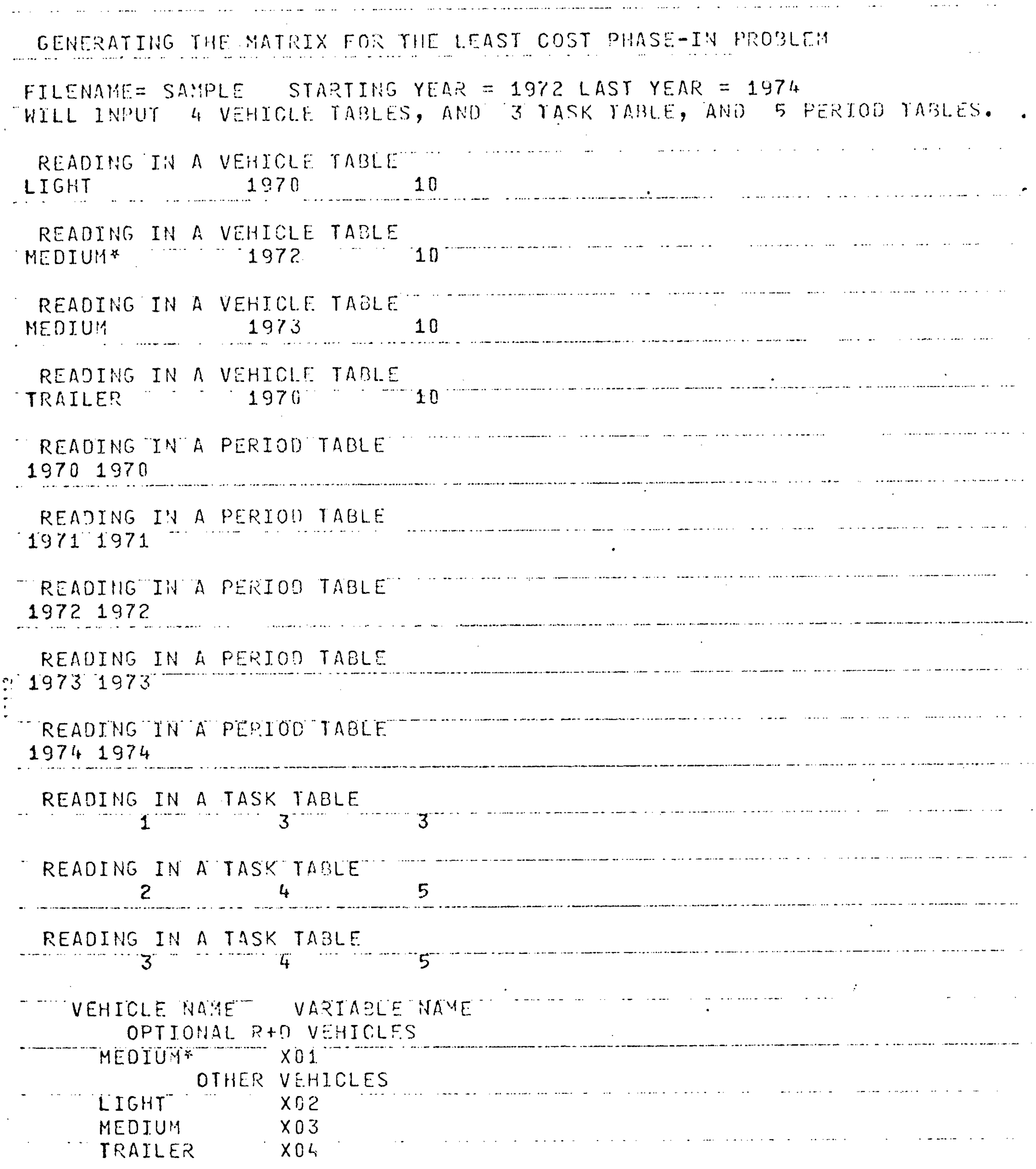
The remaining entries in the period tables should be self explanatory. The task tables are input next. Entry 33 specifies that task No. 1 will have three vehicles and three alternatives. Entries 34 through 37 input the alternative set for the first year of the planning period (compare with the figure A-2). The alternative sets for the second and third years of the planning period follow. Note that because of the introduction of the artificial MEDIUM* vehicle, the complete set of permutations yield five distinct alternatives instead of the original three. This data deck ends with an ENDTABLE card in entry 54.

We now run (process) the GENLCP program and obtain the printout of Figure A-5; parts (a) through (f). Part (a) simply prints out some input information for checking purposes, and reorders the vehicles according to the magnitude of the R&D charge; note, in this regard, that the MEDIUM* truck is "called" vehicle No. 1 (X01) since it has the R&D charge.

In the first section of Part (b), a summary of the constraint equations for this sample problem is listed; the row type (E for equality and N for free), then the row name is printed in accordance with the symbolic naming convention of Fig. 4-3. The second section of part (b), and continuing in part (c), lists the variable, the columns in which it appears, and its corresponding coefficient. Similiarly, the last section, labeled RHS, gives a summary of those rows (constraint equations) which have non-zero right-hand-sides.

Part (d) provides a cross-reference list of variable number versus variable name for use in the interpretation of the output from the BBCAV2 program. The last section of part (d) indicates that there are 25 rows and 61 columns in this sample problem. Note that in each case these are one more than was indicated previously because the cost row and the right hand side variable, respectively are now included. Finally the upper bounds, computed by the GENLCP program, are listed for the master variables; the minus sign here is superfluous.

Part (e) prints a cost summary on each vehicle along with the components of the inherited fleet. Then in the last section of part (e) and continuing in part (f), the task (alternative) tables are reproduced.



```
 GENERATING THE MATRIX FOR THE LEAST COST PHASE-IN PROBLEM

FILENAME= SAMPLE   STARTING YEAR = 1972 LAST YEAR = 1974
WILL INPUT  4 VEHICLE TABLES, AND  3 TASK TABLE, AND  5 PERIOD TABLES.

 READING IN A VEHICLE TABLE
LIGHT              1970          10

 READING IN A VEHICLE TABLE
MEDIUM*            1972          10

 READING IN A VEHICLE TABLE
MEDIUM             1973          10

 READING IN A VEHICLE TABLE
TRAILER            1970          10

 READING IN A PERIOD TABLE
1970 1970

 READING IN A PERIOD TABLE
1971 1971

 READING IN A PERIOD TABLE
1972 1972

 READING IN A PERIOD TABLE
1973 1973

 READING IN A PERIOD TABLE
1974 1974

 READING IN A TASK TABLE
        1        3        3

 READING IN A TASK TABLE
        2        4        5

 READING IN A TASK TABLE
        3        4        5

    VEHICLE NAME    VARIABLE NAME
        OPTIONAL R+D VEHICLES
      MEDIUM*        X01
            OTHER VEHICLES
      LIGHT          X02
      MEDIUM         X03
      TRAILER        X04
```

Fig A-5(a) GENLCP Output For Sample Problem

```
*NAME             SAMPLE
*ROWS
*  E   SUMX01
*  E   SUMX02
*  E   SUMX03
*  E   SUMX04
*  E   PC01
*  E   PC02
*  E   PC03
*  E   IW02PM1
*  E   IW02P00
*  E   IW04PM1
*  E   X01P01
*  E   X02P01
*  E   X04P01
*  E   T01P01
*  E   X01P02
*  E   X02P02
*  E   X03P02
*  E   X04P02
*  E   T02P02
*  E   X01P03
*  E   X02P03
*  E   X03P03
*  E   X04P03
*  E   T03P03
*  N   COST
*COLUMNS
*          (PARTIAL LISTING)
*      X01           SUMX01                -1.0000
*      X02           SUMX02                -1.0000
*      X03           SUMX03                -1.0000
*      X04           SUMX04                -1.0000
*      P01           PC01                   1.0000
*      P01           PC02                  -1.0000
*      P02           PC02                   1.0000
*      P02           PC03                  -1.0000
*      P03           PC03                   1.0000
*      W02M100       COST                   -.0008
*      W02M100       IW02PM1                1.0000
*      W02M101       COST                    .0116
*      W02M101       IW02PM1                1.0000
*      W02M101       X02P01                -1.0000
*      W02M102       COST                    .0238
*      W02M102       IW02PM1                1.0000
*      W02M102       X02P01                -1.0000
*      W02M102       X02P02                -1.0000
*      W02M103       COST                    .0345
*      W02M103       IW02PM1                1.0000
*      W02M103       X02P01                -1.0000
*      W02M103       X02P02                -1.0000
*      W02M103       X02P03                -1.0000
*      W020000       COST                   -.0015
*      W020000       IW02P00                1.0000
*      W020001       COST                    .0112
*      W020001       IW02P00                1.0000
*      W020011       X02P01                -1.0000
*      W020002       COST                    .0235
*      W020002       IW02P00                1.0000
*      W020002       X02P01                -1.0000
*      W020012       X02P02                -1.0000
```

Fig A-5(b) GENLCP Output For Sample Problem

```
 *     W020003     COST              .0342
 *     W020003     IW02P00          1.0000
 *     W020003     X02P01          -1.0000
 *     W020003     X02P02          -1.0000
 *     W020003     X02P03          -1.0000
 *     W04M100     COST             -.0025
 *     W04M100     IW04PM1          1.0000
 *     W04M101     COST              .0147
 *     W04M101     IW04PM1          1.0000
 *     W04M101     X04P01          -1.0000
 *     W04M102     COST              .0314
 *     W04M102     IW04PM1          1.0000
 *     W04M102     X04P01          -1.0000
 *     W04M102     X04P02          -1.0000
 *     W04M103     COST              .0430
 *     W04M103     IW04PM1          1.0000
 *     W04M103     X04P01          -1.0000
 *     W04M103     X04P02          -1.0000
 *     W04M103     X04P03          -1.0000
 *     P030103     X02P03          78.0000
 *     P030103     X04P03          18.0000
 *     P030103     T03P03           1.0000
 *     P030203     X01P03          24.0000
 *     P030203     X02P03          48.0000
 *     P030203     X04P03          10.0000
 *     P030203     T03P03           1.0000
 *     P030303     X01P03          50.0000
 *     P030303     X04P03          16.0000
 *     P030303     T03P03           1.0000
 *     P030403     X02P03          48.0000
 *     P030403     X03P03          24.0000
 *     P030403     X04P03          10.0000
 *     P030403     T03P03           1.0000
 *     P030503     X03P03          50.0000
 *     P030503     X04P03          16.0000
 *     P030503     T03P03           1.0000
 *     X010303     SUMX01           1.0000
 *     X010303     X01P03          -1.0000
 *     X010303     PC03              .0060
 *     X010303     COST              .0086
 *     X020303     SUMX02           1.0000
 *     X020303     X02P03          -1.0000
 *     X020303     PC03              .0030
 *     X020303     COST              .0093
 *     X030303     SUMX03           1.0000
 *     X030303     X03P03          -1.0000
 *     X030303     PC03              .0060
 *     X030303     COST              .0086
 *     X040303     SUMX04           1.0000
 *     X040303     X04P03          -1.0000
 *     X040303     PC03              .0120
 *     X040303     COST              .0070
 *RHS
 *     RHS1        PC01              .1500
 *     RHS1        PC02              .2500
 *     RHS1        PC03              .3000
 *     RHS1        IW02PM1         60.0000
 *     RHS1        IW02P00         20.0000
 *     RHS1        IW04PM1          4.0000
 *     RHS1        T01P01           1.0000
 *     RHS1        T02P02           1.0000
 *     RHS1        T03P03           1.0000
 *ENDATA
```

Fig A-5(c) GENLCP Output for Sample Problem

```
 REFERENCE LIST FOR COLUMN NUMBERS AND NAMES
C0000004-4000400004000040
          1     X01              2     X02              3     X03              4     X04              5     P01
          6     P02              7     P03              8     W02M100          9     W02M101         10     W02M102
         11     W02M103         12     W020000         13     W020001         14     W020002         15     W020003
         16     W04M100         17     W04M101         18     W04M102         19     W04M103         20     P010101
         21     P010201         22     P010301         23     X010101         24     X010102         25     X010103
         26     S0101           27     X020101         28     X020102         29     X020103         30     S0201
         31     X040101         32     X040102         33     X040103         34     S0401           35     P020102
         36     P020202         37     P020302         38     P020402         39     P020502         40     X010202
         41     X010203         42     S0102           43     X020202         44     X020203         45     S0202
         46     X030202         47     X030203         48     S0302           49     X040202         50     X040203
         51     S0402           52     P030103         53     P030203         54     P030303         55     P030403
         56     P030503         57     X010303         58     X020303         59     X030303         60     X040303
         61     RHS1

 IMPORTANT DATA ITEMS FOR INPUT TO BBCAVLP
NUMBER OF ROWS (INCLUDING COST) IS   25
NUMBER OF COLUMNS (INCLUDING RHS) IS       61
UPPER BOUNDS FOR VEHICLES IN ORDER FROM X1 THRU XN ARE
           -137.0000
           -214.0000
            -95.0000
            -51.0000
```

Fig A-5(d) GENLCP Output For Sample Problem

```
VEHICLE   VARIABLE  SAL/TRUNC   O AND M     R AND D     RETENTION   YEAR FIRST  LIFE IN
NAME      NAME      COST        COST        COST        RATE        AVAILABLE   YEARS

MEDIUM*        X01    .0060       .1400       .3000     1.0000      1972        10

LIGHT          X02    .0030       .1200      0.0000     1.0000      1970        10

MEDIUM         X03    .0060       .1400      0.0000     1.0000      1973        10

TRAILER        X04    .0100       .1600      0.0000     1.0000      1970        10

COMPONENTS OF THE INHERITED FLEET

                   1970 1971
NUMBER OF X02        60   20
NUMBER OF X04         4   -0

TASKS REQUIRED IN PERIOD FROM 1972 THROUGH 1972

  TASK 01   -  PERFORMED BY  1.00 FORCE ELEMENT(S), WITH SCALE FACTOR EQUAL 1.000
  *
   * VARIABLE    X01   X02   X04   X
     *********
  ALTERNATIVE
            1      0    67    16
            2     20    43     7
            3     42     0    12
```

Fig A-5(e) GENLCP Output For Sample Problem

```
TASKS REQUIRED IN PERIOD FROM 1973 THROUGH 1973

  TASK 02  -  PERFORMED BY  1.00 FORCE ELEMENT(S), WITH SCALE FACTOR EQUAL 1.000
  *
   * VARIABLE     X01   X02   X03   X04   X
     *********
  ALTERNATIVE
             1      0    69     0    17
             2     22    45     0     8
             3     45     0     0    13
             4      0    45    22     8
             5      0     0    45    13

TASKS REQUIRED IN PERIOD FROM 1974 THROUGH 1974

  TASK 03  -  PERFORMED BY  1.00 FORCE ELEMENT(S), WITH SCALE FACTOR EQUAL 1.000
  *
   * VARIABLE     X01   X02   X03   X04   X
     *********
  ALTERNATIVE
             1      0    78     0    18
             2     24    48     0    10
             3     50     0     0    16
             4      0    48    24    10
             5      0     0    50    16
```

Fig A-5(f) GENLCP Output For Sample Problem

The reader should note that very little near new information is produced by the GENLCP program — for the most part, GENLCP merely formats, reorders, checks, and performs bookkeeping operations in preparation for entry to the BBCAV2-REPGEN algorithm.

The first data deck for the BBCAV2-REPGEN algorithm is prepared as illustrated in figure A-6. We first assign a solution name in entry 1. The first, second, fourth and fifth fields of entry 2 are omitted as described on page 4-26. The third field in entry 2 indicates that there are four concave cost functions. The zero in the sixth field suppresses printing of the subroutine calls; the 1 in field No. 7 prints a listing of the primal iterations of each linear program; and the 1 in field No. 8 prints the entire set of LP solutions. Fields 9 and 10 are the standard specifications for the size of the array BLIST. The 1 in field No. 11 prints the column numbers and their corresponding values for each node. The last field, set to 20, establishes the limit on the number of nodes that will be evaluated prior to termination.

Entry 3 has four fields which establish (1) a tolerance factor of 0.005 (i.e., the solution will be within one-half of one percent of the theoretical optimum), (2) a program time limit of 90.0 seconds prior to termination (the solution to the sample problem actually used only 48 central processor seconds), (3) that no initial solution (basis) will be input, and (4) that we wish to obtain a detailed output. The second data deck for the BBCAV2-REPGEN algorithm is prepared as illustrated in figure A-8. As discussed on p. 4-35, the REPGEN data deck is very easy to prepare since most cards are duplicates of the GENLCP data deck. After the title card, the vehicle tables are inserted with cards of type 2 deleted. The period tables come next using only the header cards and cards of type 1. Note that the period designators have been inserted on all cards of type 1 in columns 11 and 12. The ENDTABLE card in entry 24 ends the data deck.

When the BBCAV2 program in the BBCAV2-REPGEN packet is loaded and processed, the printed output gives complete information vis-à-vis the optimal solution as well as all intermediate nodal solutions. The printout is long and involved and is in a coded format. The REPGEN program in the BBCAV2-REPGEN packet will decode the BBCAV2 output and

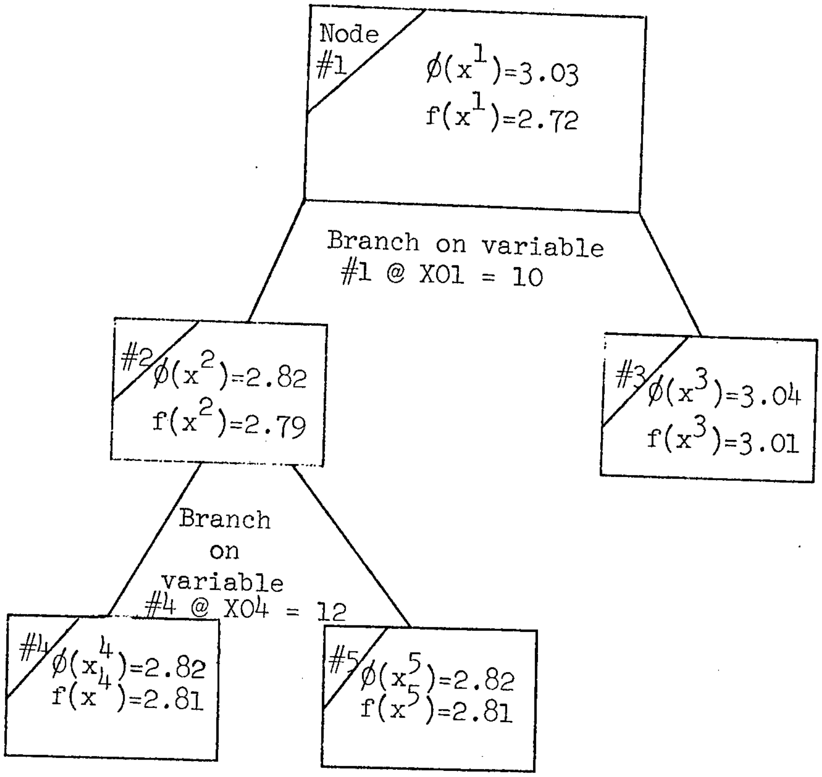
```
1 SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL
2                  4                 0     1     1     25   131    1   20
3        0.005        90.0         0.0         1.0
```

Fig. A-6 The Sample Problem Data Deck For the BBCAV2 Program

present all essential information, hence, we neither present nor discuss the BBCAV2 output. After some experience with these programs, the user may suppress all intermediate information if he so desires. We do present here a branching tree (like that of Fig. 2-2b) to lend further clarity to the intermediate solution.

The "tree" of intermediate solutions (see Fig. A-7) is illustrative of the iteration process of the branch-and-bound algorithm. Node #1 represents a complete linearization of the non-linear problem and establishes the first reference solution, $\phi(x^1) = 3.03$ million dollars. A lower bound to all solutions, $f(x^1)$, is also determined and equals 2.72 million dollars. The branching rule is then applied and variable #1 is selected, at the branching value of X01 = 10. The linear programs associated with node #'s 2 and 3 are then evaluated. Node #2 yields a better reference solution of $\phi(x^2) = 2.82$. Node #3 is found to contain no better solution than $\phi(x^2)$ because $f(x^3) > \phi(x^2)$, hence node #3 need no longer be considered. The next best branching variable is #4, at a value of X04 = 12. The linear solutions to nodes 4 and 5 yield identical results, indicating a solution at the bound. The process terminates here because the smallest lower bound is within one-half of one percent of the current reference solution. Detailed information regarding the optimal (and final) solution is obtained through the REPGEN program.

REPGEN in the BBCAV2-REPGEN algorithm is then processed resulting in the output of figure A-9, parts (a) and (b). Part (a) contains an overall COST INFORMATION summary together with a breakdown of the number of PURCHASED RESOURCES (vehicles). Part (b) illustrates a breakdown of the STORED (mothballed) RESOURCES and the TOTAL RESOURCES USED by period. From these tabulated results, we have constructed a bar chart in figure A-10 to better illustrate the optimal solution. In this display, the results of a cost minimization over time are illustrated by a bar for each year and each vehicle. The height of the bar is a measure of how



$\phi(x^i)$ = the actual (non-linear) solution for node i.

$f(x^i)$ = the lower bound (linear) solution for node i.

Figure A-7 A Branching Tree for the Sample Problem

```
SAMPLE        1972 1974      4       3       5
VEHICLE
MEDIUM*   1972          10
        .006       .14          .30          1.0          .006
VEHICLE
LIGHT     1970          10
        .003       .12          3.0          1.0          .002
VEHICLE
MEDIUM    1973          10
        .006       .14          0.           1.0          .006
VEHICLE
TRAILER   1970          10
        .010       .16          0.           1.0          .012
PERIOD
1970 1970 01
PERIOD
1971 1971 00
PERIOD
1972 1972 01.15
PERIOD
1973 1973 02.25
PERIOD
1974 1974 03.3
ENDTABLE
```

Figure A-8 The Sample Problem Data Deck For The REPGEN Program

many vehicles were selected — the color black indicates vehicles existing or retained — a dotted section indicates vehicles purchased — a blank section indicates vehicles stored (mothballed) for later use. One can observe the trend toward a fleet of only medium trucks and trailers (perhaps because of the high labor costs of operating so many light trucks). The MEDIUM* trucks are not chosen for 1972 because of the high purchase cost for early delivery. The slack is taken up by a large purchase of trailers in this first year; the trailers are needed in the later years anyway. Storage of a few trailers is indicated in 1973,

```
                    SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

COST INFORMATION

                *             *             *             *             *
                *   R AND D   * PROCUREMENT *  OPERATING  *   SALVAGE   *     TOTAL
**************************************************************************************
                *             *             *             *             *
PERIOD  01      *             *        .112 *       1.060 *        .019 *      1.152
                *             *             *             *             *
PERIOD  02      *             *        .199 *        .856 *        .026 *      1.028
                *             *             *             *             *
PERIOD  03      *             *        .034 *        .956 *        .001 *       .989
                *             *             *             *             *
                *             *             *             *             *
TOTAL           *       0.000 *        .345 *       2.872 *        .047 *      3.170
**************************************************************************************


TRUNCATION VALUE FOR RESOURCES =         .348
```

```
                    SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

PURCHASED RESOURCES

                *             *             *             *
                *   MEDIUM*   *   LIGHT     *   MEDIUM    *   TRAILER
*************************************************************************
                *             *             *             *
PERIOD  01      *       0.000 *       0.000 *       0.000 *      12.000
                *             *             *             *
PERIOD  02      *       0.000 *       0.000 *      42.667 *       0.000
                *             *             *             *
PERIOD  03      *       0.000 *       0.000 *       7.333 *       0.000
                *             *             *             *
TOTAL           *       0.000 *       0.000 *      50.000 *      12.000
                *             *             *             *
```

Fig. A - 9(a) REPGEN Output For Sample Problem

```
                SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

STORED RESOURCES

             *          *          *          *
             * MEDIUM*  * LIGHT    * MEDIUM   * TRAILER
*********************************************************
             *          *          *          *
PERIOD  01   *    0.000 *    0.000 *    0.000 *    0.000
             *          *          *          *
PERIOD  02   *    0.000 *    0.000 *    0.000 *    2.793
             *          *          *          *
PERIOD  03   *    0.000 *    0.000 *    0.000 *    0.000
             *          *          *          *
```

```
                SAMPLE---SIMPLE TRUCK PROBLEM---OPTIMAL

TOTAL RESOURCES USED

             *          *          *          *
             * MEDIUM*  * LIGHT    * MEDIUM   * TRAILER
*********************************************************
             *          *          *          *
PERIOD  01   *    0.000 *   67.000 *    0.000 *   16.000
             *          *          *          *
PERIOD  02   *    0.000 *    3.578 *   42.667 *   13.207
             *          *          *          *
PERIOD  03   *    0.000 *    0.000 *   50.000 *   16.000
             *          *          *          *
```

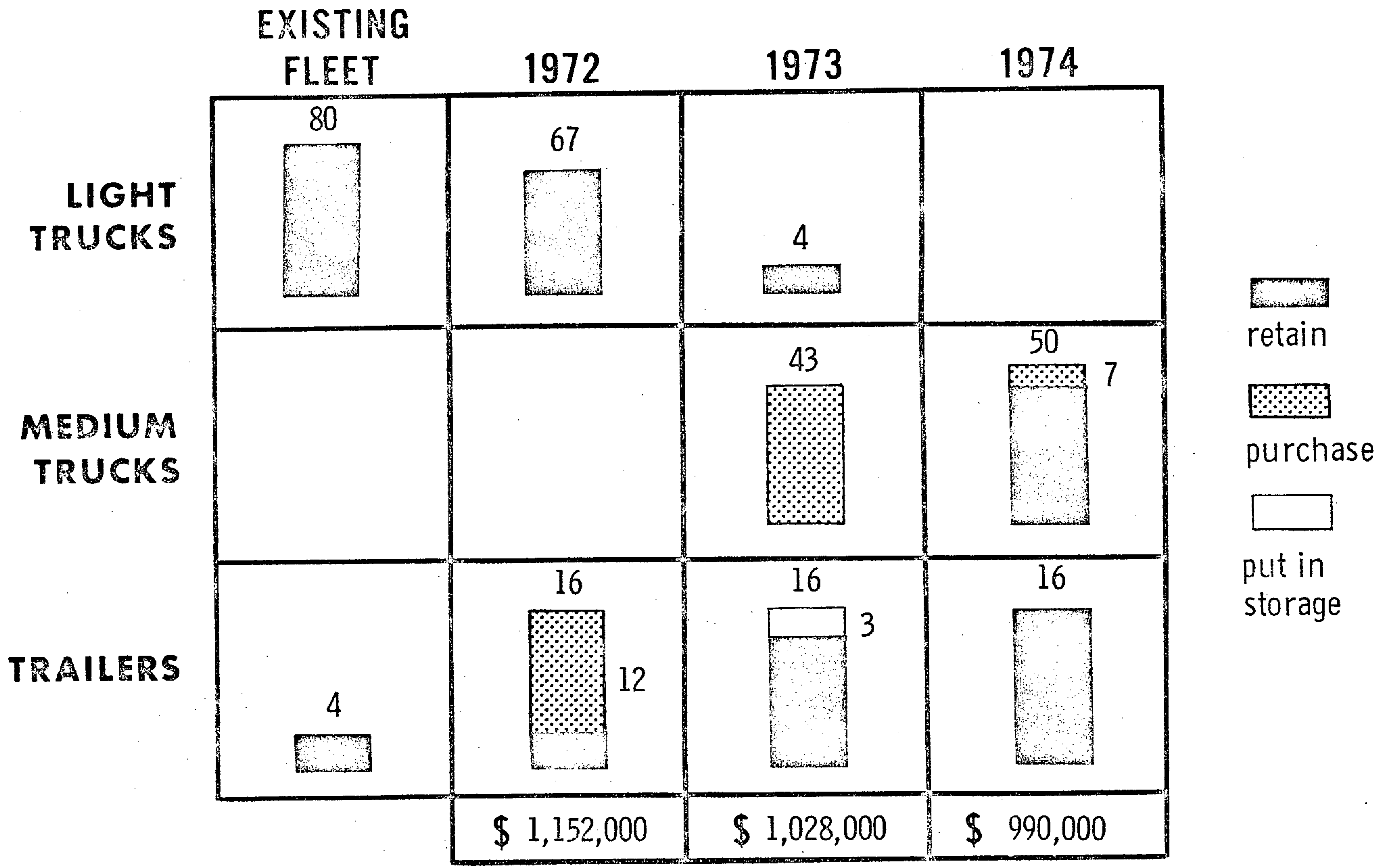
Fig. A - 9(b) REPGEN Output For Sample Problem (perhaps a surprising result but, under the circumstances, a reasonable one since buying medium trucks or retaining a larger number of small trucks for this sub-period are very costly alternatives). Finally, one can observe the relatively high start-up costs due to the purchase of the entire trailer fleet in 1972, and then most of the medium truck fleet in 1973. Nevertheless, these high start-up costs yield the least total cost over the planning period.

The COST INFORMATION summary indicates a total real cost (that which must be allocated) of 3.17 million dollars. This differs from the branch-and-bound solution value of 2.82 million dollars by the truncation value; i.e., it is deemed proper to include the truncation credit for purposes of optimization, but this dollar credit (unlike salvage) is not considered to be available for other purposes. From the procurement cost summary, it would appear that none of the procurement cost constraints were binding; however, the reader should recall from p. 4-36 that these data were calculated from the linear procurement estimates and, upon close inspection of the BBCAV2 printout, one could observe a binding cost constraint in the second period. As previously intimated on page A-7, this relatively poor correlation between linear estimate and actual value (in this case in the medium vehicles) can give erroneous results.* For cases in which the discrepancy is excessively large (judged not so in this sample problem) the programs should be reprocessed with a better linear estimate. The restriction of available funds in period two is <u>probably</u> the cause of the retention of the light vehicles, in lieu of the purchase of the full complement of medium vehicles for that period. Finally, one should observe that the RESOURCE summaries in general contain non-integer values — a consequence of the continuum of solutions to linear programming problems. It is incumbent upon the user to provide a physically meaningful interpretation to such results — as we have done for this sample problem in figure A-10.

* The difficulties introduced here due to a poor linear estimate can be avoided in the future by employing a recently developed modification to the current algorithm.[5]

Figure A-10

A Sample Optimal Plan for the Ace Trucking Company



TOTAL COST $ 3,170,000

APPENDIX B

SUBROUTINE DESCRIPTIONS

APPENDIX B

MATRIX GENERATOR ROUTINE DESCRIPTIONS

GENLCP - reads and analyzes input data, creates column names and row names, determines non-zero values of the matrix of coefficients, creates the MPS360 file, and outputs the documentation listing.

YRCOST (J) - has parameter J, which is vehicle number, and determines for this vehicle all cost information (see Chapter 4 for detailed description).

YINTERP (NVR, NTR, NRY) - determines for all tasks (NTR) which of the NVR vehicles will not have been developed by the year NYR, and eliminates from those tasks all alternatives in which the "non-existent" vehicles are accomplishing something which could be done by an existing vehicle.

MATFILL (N, M) - creates from the MPS360 file the file for BBCAV2 which is an N-row by M-column matrix of coefficients, and also creates the reference list for matching column numbers and names.

MAIN PROGRAM ROUTINE DESCRIPTIONS

BBCAV2 - is the programmed implementation of the main logic structure of the branch and bound algorithm; selects node from branching tree, branches on it defining two new nodes, and determines when optimality has been achieved.

BOX1 - defines the initial node of the branching tree and establishes the problem framework in which the main program will iterate.

INITA (NCF, N, M) - reads the matrix file from tape and stores it on disk in the form acceptable to the LP; the parameters N and M define the number of columns and rows in the matrix, and the parameter NCF is the number of columns having nonlinear cost functions.

GETPHI (KFX, XPHI, PHI, SUMPHI) - evaluates the nonlinear cost functions (see Chapter 4 for a detailed description).

TABOUT (IRT) - outputs the general information concerning the node being evaluated, and if IRT = 1, also prints the nodes on the branching list.

READIN - transfers the input basis to the file which the LP uses for storing its current basis on.

NXBRN (XT, SIGMAT, NXB) - determines the best branching candidate, NXB, given the present X-vector, XT, and the node information, SIGMAT.

TIMEC - determines how long the program has been running, prints the time or interrupts depending on whether or not this time is less than the input maximum.

GETASQ (NOES, ELM, JSQ) - orders the elements of the vector ELM, of length NOES, in ascending sequence, keeping the index variable, JSQ, associated with the vector in the corresponding sequence.

GETC (KCX, BLT, ULT, CT) - determines the slope of a straight line on the cost function from the lower bound, BLT, to the upper bound, ULT, for the KCX variable and stores this in the cost vector, CT.

PRESET - initializes core storage at beginning of the algorithm.

SET (TMMAX) - initializes the time limit which is used by the routine TIMEC by adding the limit, TMMAX, to the clock time.

PARAMS - reads and stores the information on the parameter and bound cards of the input deck.

LP

LP is the linear programming system driving subroutine which directs the overall solution stages through:

SETUP - which initializes all data for the A matrix files and the solution bookkeeping.

MAPIN - which introduces any prior solution known for the problem.

INVERT - which solves the problem equations to generate the current solution represented by the basis inverse and the values of the basic and key variables in the current solution, or an artificial solution.

PRIMAL - which solves the linear programming problem.

MAPOUT - which stores the solution found and loads it into the output vectors IX and X.

LP begins with the overall common definitions for the LP system, and must be loaded first since the common statements /A/, /B/, /CORE/, /ROWTYP/, /IXX/, /XX/, /NAMES/ overwrite the smaller dimensions specified in later subroutines which can remain unchanged regardless of problem size.

AJ contains the operating core columns and the complete basis inverse B at AJ(IØRG) ≡ B(IORG), see below.

The first stage is to specify the A matrix files IA1 - used for the A matrix less GUB rows, INPUT - used as the source of the unpacked A matrix by columns, witten in binary, one column per record, and IMAP - used by MAPOUT, MAPIN and INMAP as a file for the BCD MAP cards defining the basis, initial and/or final.

Files IA1 and IA2 are specified as blocked. Meaning that each physical disc write or read is of as many columns as the buffer sizes for IA1 and IA2 will allow, and not just one column, as actual written in FORTRAN. This considerably reduces the disc access time denoted as PP time on the CDC 6400.

NWAJ, the number of words in AJ is used to compute the number of columns available in core.

The second step is to initialize the LP calling parameters with the LP calling arguments. The matrix generator locates the cost row

ICOST as the last row MROWS which is INPUTM for the linear program common/INPUT/. The right hand side is JRHS, placed as the last column

NCOLS which is INPUTN for the common/INPUT/. The number of bounds NBDS is the number of changes NCHGS, the calling parameter, and the first NCHGS columns are bounded by BBCAV convention. The values of the bounds are in UBS.

The second stage ends with specification of LP system print and termination controls.

LP Cut-Off

STATUS terminates the program if any linear program takes longer than TMAX seconds or K5 iterations ITRN. The termination causes a MAPOUT allowing restart of the linear programming system but not necessarily BBCAV.

Diagnostic Snapshots

A snapshot of the current solution and column format can be had from XCHECK by setting K4 to

$$K4 = 1000 \times N1 + N2$$

giving a snapshot of iterations N1 through N2 inclusively. K4 = 0 suppresses XCHECK. The format is explained in the XCHECK subroutine writeup.

Print Control

This is achieved by K3.

K3 = 0 prints everything

K3 = 1 prints no LP system output except error messages.

Other values of K3 give a selective print of all or some of the respective outputs according to the prime factors of K3.

Specifically K3 should be a product of primes and

$$K3 = 2 \times 3 \times 5 \times 7 \times 11 \times 13$$

gives all print options. To obtain selective control:

1. To print the MAPIN cards as read K3 has factor 3.
2. Inversion diagnostic data from INVERT on infeasibilities of the current solution as found, the columns rejected during

an inversion or reinversion and the final infeasibility if any, K3 has factor 13.

3. MESSG prints linear programming system verb entry names and entry times and serveral messages if K3 has factor 7.
4. STATUS prints the status of the PRIMAL iterations at beginning and end of K3 has factor 11.
5. MAPOUT places a basis inverse B, IBASIS, KEYS and BETA representing Restart data sufficiant to avoid an initial invert on file INPUT if K3 has factor 2.
6. MAPOUT prints the solution status in packed format when called if K3 has factor 5.

Thus to obtain printouts of inversion diagnostics, a mapout, verb entry times and status data set K3 = 5 x 7 x 11 x 13.

The third stage of the program LP calls the system verbs listed earlier.

The basis inverse is at B(IØRG) where IØRG is M^2 words down from the end of AJ, i.e. NWAJ. The remaining space in AJ is allocated to columns of which NCRMAX can be fitted in. There must be at least 5 columns slots available, three for CHECK to retain columns and two for work space. Fifty columns are recommended

Finally MAPOUT moves the current variable state to file IMAP, the solution to INPUT, the packed variable values to IXX and XX. IXX has the indices in ascending order of non-zero variables, and XX has the corresponding values. There will be between INPUTM and INPUTM + NBDS non-zero values followed by zeros.

Variable Lengths

In /IXX/, /XX/ IXX, XX should be set to 100 or INPUTM + NBDS if larger.

In /CORE/ AJ should be big enough to take the basis inverse $(\text{INPUTM}+1-\text{L})^2$ words plus 10 to 100 columns at (INPUTM+1-L) words each where L is the number of GUB rows.

In /ROWTYP/ IROWTP should be 100 or INPUTM+1 of larger than 100.

In / NAMES/ NAME should be 100 or INPUTN+1+S if larger, where S ≤ INPUTM is the number of inequalities and free rows.

In /A/ ALPHA should be 100 or INPUTM+1 if larger.

In /B/ BETA should be 100 or INPUTM+1 if larger.

All other commons are correctly sized in LP.

EXISTS

All exists from primal are via the subroutine EXISTS for the purpose of user parameter settings.

SETUP

This subroutine is the system verb which has the task of initiating the LP system when starting from scratch.

SETUP first of all initializes all LP system parameters, then examines the row types constructing a logical or slack column for each nonequality row and writes these to disc using calls to OUT of IO. An extra free row is incorporated for the phase 1 cost row and the logical columns for free rows are marked basic.

SETUP then reads the A matrix columns from the binary INPUT file and writes then out to disc using calls to OUT of IO. For each column the NAME vector is set to record the column type (free/null), the column GUB packet number or zero and the column bound index or zero. The right hand side vector is recorded in core in RHS.

Finally, SETUP rewrites IBASIS and the RHS vector to place the GUB row elements at the end. The count of GUB rows is recorded in L and the actual row count is reduced by L. The cost row marker ICOST is reset to its new position in the rearranged rows.

IO

This subroutine handles all disc to core transactions and keeps track of column bookkeeping.

OUT writes two files of columns of the A matrix writing one column in each file per call. The first file IA1 contains columns less their GUB elements. The second file IA2 contains columns less their GUB elements and any zero elements and is written in a packed format.

IN reads file IA1 cyclically up to NT times, in search of a particular column rewinding when appropriate. It is normally accessed for sequential columns by CHECK but INVERT uses it to locate basis, at-bound and key columns marked in the NAME vector.

INPCKD reads file IA2 cyclically up to NT times in search of a column rewinding when appropriate. It is only accessed in random forward increments searching for key columns and thus uses a packed file.

After NT reads, sufficient to locate any column, both entries cause an error message and dump.

Once IN or INPCKD have located the required column and read it to a slot in AJ( ), the column index is loaded to the corresponding position in JA, its reject memory in JAREJ is cleared and its mnemonic (unused) is placed in JAK.

Thus it is not possible to read a column into core without adjusting the bookkeeping of what is in core.

MAPIN

This subroutine is the system verb which sets the bookkeeping of the column status and allows a restart from a previous status. It is designed to read a file IMAP generated by MAPOUT and loaded by INMAP to file IMAP.

MAPIN reads settings of NULL, BASIC, KEY and ATBND designated at random one type per card up to 4 columns per card for each type. Each type sets the column status marker in NAME appropriately.

Restarts

MAPOUT writes the LP system status onto the end of the INPUT tape file when MAPOUT is called, and provides an INVERSE card for MAPIN use. The INVERSE card causes MAPIN to check for an inverse plus bookkeeping data and the solution status on the INPUT tape, and read it if present.

INMAP

Is the entry designed to read the input card stream for MAPIN cards and load them onto file IMAP. It is terminated either by an end of file or and END card.

Manual preparation of MAP cards is possible and extensive checks in MAPIN will detect and avoid most errors.

INVERT

This is the system verb which inverts or reinverts the current basis as defined in NAME records and completes the basis with artificials.

When INVERT is called, an inversion occurs only if the current iteration exceeds ITNINV. When it does ITNINV is increased by INVF, and an "INVERT" message is printed.

INVERT first clears the basis records and the GUB packet basis column count, sets up a unit basis and for each GUB row without a key chooses the first valid GUB packet column as key. It then cycles the column status records in NAME until it locates a basic, key or at-bound column, which is retrieved by IN. Key and at-bound columns are accumalated in GAMMA scaled by their packet righthand sides (if keys) or their bounds. If basic columns are in a GUB packet, the key is located by INPCKD, subtracted from the column and the result transformed and pivoted into the basis in a row determined by PIVOT.

When all NAME records have been checked and the columns incorporated or rejected by PIVOT, the basis record is completed with logicals or if necessary with artificials. The artificials are then constructed in DELTA transformed and pivoted into the basis. Finally, FEASCH is called to construct and check the solution feasibility.

NB. The MAPIN used can be partial, complete, redundant or nonsense.

FEASCH

FEASCH is called by INVERT to compute $\beta = B^{-1}\gamma$ given γ in GAMMA and B^{-1} in B. The resulting β elements in BETA are checked for feasibility and the basis is adjusted if infeasible until the resulting BETA is feasible and the phase IPHASE is 1 or 2.

The method is to cycle each element of BETA from 1 to M, compute it, check if it exceeds a bound then check if it is positive. If it exceeds a bound, the basic column is set "at-bound," and the bound is subtracted from that BETA(I) which then becomes negative and infeasible. If it is negative i.e., infeasible, its sign is reversed and the column is replaced by its negative artifical*, to pick up the infeasibility directly, (the artificial need not be transformed) and pivoted into the basis in place of the old basic column. Finally, if GUB rows are present the last L entries in BETA are filled with the values of the key variables.

Feasibility of the keys is maintained by calling KEYCH to move the infeasible key (essential packet) to a basic position in a non GUB row and processing it as above as an infeasible variable.

If any infeasibilities have been encountered, or the resulting Phase 1 cost is larger than CTOL, Phase 1 initiates otherwise Phase 2.

* The negative artificial of a column A_i is $-A_i + e_m$ is the mth column of the identity matrix. The negative artificial of an artificial $e_i + e_m$, is $-e_i + e_m$.

PRIMAL

This subroutine is the main LP verb which solves the LP problem phases 1 and 2.

PRIMAL notes its entry and time using MESSG, then picks up the cost row for its current phase 1 or 2, the appropriate π row in the inverse and sets the phase 1 row to free in phase 1 or equality in phase 2.

The basic solution cycle is counted by ITRN. If ITRN exceeds K5 or if CP time exceeds TMAX a MAPOUT is called by STATUS followed by EXIT.

The solution cycle proceeds with COLUMN to find an in-core column JCOL. If JCOL = 0 no column is found and the phase terminates. If the cost is zero in phase 1 this is the feasible solution termination, if phase 2 this is the optimal solution, if non-zero in phase 1 there is no feasible solution.

Next ROW is called for a pivotal row IROW. If IROW = 0 no row is found and the problem is unbounded.

Next the pivotal element is checked for size and degeneracy. If it is too small NREJ is indexed. If 5 bad columns have occurred an INVERT is called to check the inverse. If more than 100 bad columns have occurred the problem terminates either in phase 2 as optimal, or with a dump. If the pivotal element is okay, the cost change THETA * DJ(JCOL) is checked. If this is smaller than CTOL, NDEG is indexed. If more than NDEGLM degenerate columns have occurred and there are no more good columns the column is accepted. Otherwise in either case the old column is rejected and a new column is selected by COLUMN ignoring the previous selections.

Next the step is saturated to the bound on the column. If it exceeds the bound and the column is not at bound, the column is set ATBND. If the column is ATBND, the column is set free. In either case there is no pivot and the solution is corrected for the bound change but there is no basis change correction and NREJ = 1 to suppress pricing in the next iteration.

If the step is within the bounds, a basis change will be made. The rejected column is located first in the basis of IROW ≤ M, then in the keys of IROW > M. If IROW ≤ M there is no key change, the new column is pivoted in by PIVOT at IROW. If the new column is AT BND the step is off the bound and the new column value EPSI is corrected to the bound value less the step. Then the new column is made basic, the rejected column is made free, and the solution step made and the new basic column value set to EPSI.

If IROW > M there is to be a key change. If the GUB packet is essential, it has other basic columns and the key is changed for one of these using KEYCH, then IROW ≤ M and the previous case follows.

If the GUB packet is not essential it has no basic columns, so the key is changed to the new column and the old key is dropped. The new key value EPSI = THETA and a normal step is made as before without a pivot and pricing in the next iteration is suppressed.

After every pivot the rejected columns are cleared.

At the end of the iteration cycle STATUS reports the solution change.

After every row and column selection, or at any optimality stage, XCHECK is called for a debug which occurs if K4 is set > 0. See LP for details.

STATUS

STATUS prints out the status of PRIMAL iterations every cycle under the headings.

PHASE = (IPHASE) - the LP phase 1 or 2.

ITER = (ITRN) - the LP iterations count.

TRY = (NTRY) - the number of iterations with the same set of columns in core + 100 x maximum number of tries.

VAL OBJECTIVE = (BETA(IC)) - the solution value of the current cost row.

NDJS = (NDJS) - current count of negative DJ's an estimate of non-optimality of the current core columns.

NARTS - the current number of artificial vectors present.

VALUE DJ IN = (DJ(JCOL)) - the value of the DJ for the column chosen to enter.

COL IN = (JPØS) - the internal number of the column chosen to enter.

CODE = (NAME(JPØS)) - the status of this column.

COL OUT = (JOUT) - the column rejected.

CODE = (NAME(JOUT)) - the status of the column.

NSCAN - the current number of rewinds of file IA1 the A matrix plus the number of columns active in core, or columns read on disc.

Note: If JOUT is zero, no column was rejected and its code is zero.
If JCOL is zero or IROW is zero, these are taken as termination markers and the NOTE obtained in the STATUS call is printed e.g. PRIMAL--END, etc...
If JOUT > NT artificial code is constructed equal to the 10^9 x IROW.

ROW

This subroutine is called by PRIMAL to locate the pivot row IROW in the selected column JCOL.

ROW first transforms the in-core column JCOL to the current basis representation in ALPHA, reconstructing the complete column including GUB elements which occupy the last L positions.

The row selection depends upon whether the column is at-bound or not, for if at bound the column represents the slack vector and the step is negative. For either case the minimum THETA is found which

(i) drives the resulting solution to zero or

(ii) drives the rejected column out at bound, depending upon the sign of the potential pivot element ALPHA(I).

These are case 2 and 3 for a normal column and cases 3 and 2 for an at bound column. Upon exit THETA is the step in row IROW, core-column JCOL, (JPØS on disc) and ITYPE is 2 or 3 for the type of step.

If no row is found IROW = 0, ITYPE = 1 indicating an unbounded step and THETA = 1·E35.

COLUMN

This subroutine locates a potential column entry JCOL from those in-core, or calls CHECK to search all or part of the disc for more columns and uses these.

COLUMN counts NTRY selections with the current columns. If more than NCRMAX, or no columns exist in core it locates up to NCRMAX new columns with a call to DISC. If no columns are found the problem is optimal and JCOL = 0 at exit.

The in-core columns are then priced out, unless no pivot has occured (NREJ or NDEG ≠ 0) because of column rejection or DISC has just been called. PIKEY is always set to the current key price for the packet recorded in JPKTO. If a column is in a packet, its price is adjusted for the key price. All columns are priced apart from rejected columns.

The best unrejected column is now found by searching the DJ values. At bound columns have DJ reversed as they correspond to the slack column, and the number of negative DJ's is counted in NDJS.

If no good column is found, i.e. the best DJ is above the DJTOL threshold, DISC is called to search for more columns unless these columns are new,denoted by NTRY = 0 (no selections with these columns).

Upon exit JCOL is the in-core location of the best column found in-core (or from disc) called JPOS, or JCOL = 0 denoting no column. If JCOL = 0, JNCORE is reset to the number of columns in core (because DISC has deleted the count of columns that were there) in order to try to save a disc read in the next phase if any.

DISC

This subroutine checks the disc for more columns and selects those which are currently "not bad."

First DISC calls INVERT to see if the iteration count ITRN has exceed the next invert point and inverts if necessary.

DISC reads the columns in batches of NBCH columns serving 1 column/batch. If fewer than NCRMAX columns are actually used these are read directly into core where they stay, once and for all. Alternatively the current file IA1 position JNT is found by INPOS and DISC examines the columns starting at JNT + 1, proceeding cyclically, changing batch every NBCH columns. If the new packet number PKT is different and nonzero the new key is located and read over any unused old key, or into the next vacant AJ slot, by INPCKD.

The column type is found in JTYPE and null (0), basic (2) and key (4) columns are skipped. Free (1) and at-bound (3) columns are read by IN to the next location JORG. The new column is priced out correcting for its packet if ≠ 0, and if the new price DJNEW is worse (≥) than DJOLD (the best of the current batch) the column is skipped. Otherwise this column is preserved as IORG in the batch records and the best batch column DJ as DJOLD.

Every NBCH column, column IORG is saved if it is better than DJTOL and IORG is reset to the next vacant column.

DISC will work if the packets are disjoint, (separated by zero packet columns), and also if the packets are mixed up, (alternate columns in different packets) but with much loss of efficiency due to multiple key searches and rewinds of file IA2.

DISC always pulls in the key of each packet first for each packet, using the packed file IA2 regardless of where the key is located in the packet.

Subroutine KEYCH

Changes the key for an essential GUB packet, to one of its basic columns in the packet, selecting the first one. The basis inverse, B, solution in BETA and current column in ALPHA are corrected for this rearrangement.

Subroutine SETBND, SETBNB, SETNNN, SETKEY

Sets and unsets the state of a column J in NAME (J), to either free (1), null (0), basic (2), at-bound (3) or key (4), respectively.

Function DOT, DOTS

Computes the inner product x'y in either double and single precision, respectively.

Subroutine MAPOUT

Writes the states of the null, basic, key and at-bound columns onto BCD cards, placed on file IMAP, and also places the current inverse B solution BETA and basis bookkeeping IBASIS and KEY onto the end of input tape INPUT to allow instant RESTART.

Function BOUND

Returns the value of a column bound, if bounded, or 10^{70} if unbounded, or artificial.

PIVOT

This subroutine pivots a new column ALPHA into the basis inverse B at row IROW. If IROW is zero the best pivotal row is found.

IROW is zero the basis is checked for empty slots or slots containing the column disc index JPOS. If the latter is found, this row is used as IROW since this is SETUP's method of fixing logicals for free rows. For null basis entries PIV and IROW track the largest ALPHA element and its row, and this is used as the pivot element unless it is less than PIVTOL where upon the column is dropped with IROW = 0 as a marker.

If IROW is non-zero, the pivot ALPHA (IROW) is checked against the PIVTOL for possible errors. If the pivot is not unity, the inverse row IROW is normalized by the pivot. Then for every non-zero ALPHA entry at I, that multiple of the inverse pivot row is subtracted from the Ith inverse row (skipping the pivot row).

KEYFND - function

KEYFND find the location in core of the key column for the specified packet. If none is found in core it returns a value zero.

If the calling argument is zero KEYFND locates any key in core which has no associated GUB packet columns. If no key is found in-core it returns a value zero.

Otherwise the value of KEYFND is the column location 1 to NCRMAX.

ESCAPE

ESCAPE causes termination with a snapshot of the working core followed by a call to file 0. This will generate an abort condition suitable to generate a system dump. If it is desired to do this, use:

```
DEBUG.
LGO.
EXIT.
DMP (LP, ESCAPE)
7
8
9
```

This will dump the core using the labelled system dump from subroutines LP to ESCAPE, which should be first and last respectively.

Consult the variable list for a definition of the global variables.

All calls to ESCAPE are preceded by an ERRØR message of explanation of the fault condition.

XCHECK

XCHECK delivers a core snapshot if ITRN lies between N_1 and N_2 where $K4 = 1000\ N_1 + N_2$. ($K4 = 0$ suppresses XCHECK.)

XCHECK prints using the following format.

(a) Col 1 indexes the normal and GUB rows respectively.

(b) Col 2 prints the basis IBASIS and KEYS respectively.

(c) Col 3 is the current column representation ALPHA of JCOL.

(d) gives the pivot position IROW.

(e) Col 4 gives the current basic and key variables respectively.

(f) step is THETA the proposed step, before bounding.

(g) the column bound.

(h) the selected column disc index.

(i) is the list of core-column disc indices.

(j) Col 5-14 is a list of 10 columns around the selected column in their current basis representation.

(k) is a list of the column name codes at the XCHECK instant.

REPORT GENERATOR ROUTINE DESCRIPTIONS

REPGEN - the main program acts principally as a control program calling other routines to perform specific functions; determines if all solutions have been interpretted and initializes storage for each solution.

SETUP - reads input deck and reference list file into core storage.

INSOLN - interprets the meaning of each column in the solution and stores its value in the appropriate array(s).

YRCOST (J) - same as in matrix generator.

VALUES (N, ISTART, IEND, VAL) - determines cost information associated with each "X" or "W" type column in the solution; N is the number of the vehicle type, ISTART is its first year of existance, IEND is its last year of existance, and VAL is the number of those vehicles.

CINFO - this routine organizes, tabulates and outputs the table of cost information.

PINFO - this routine merely formats and outputs the last three tables of information; purchased resources, stored resources, and total resources used.

APPENDIX C

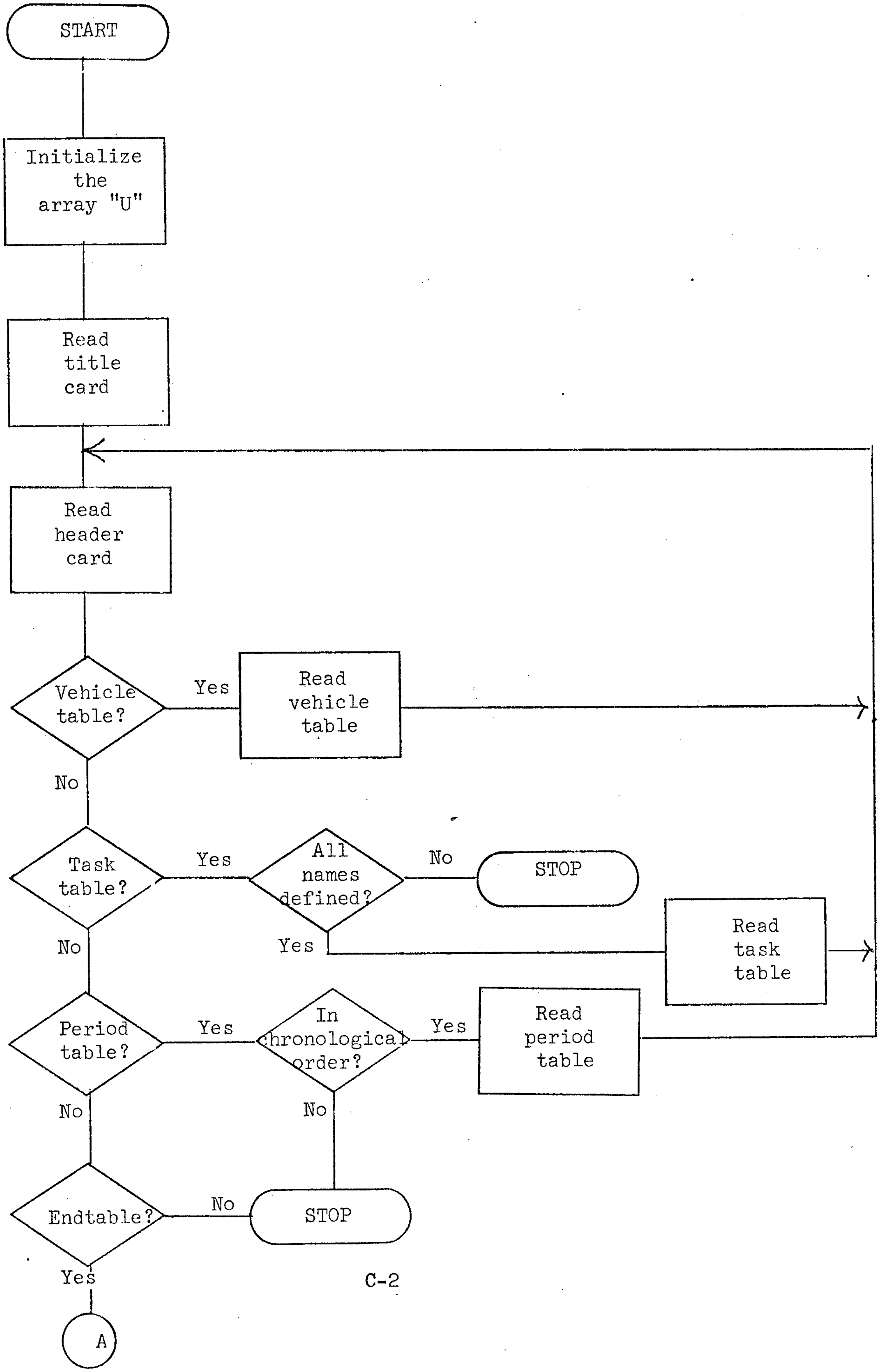
"I·OWCHARTS

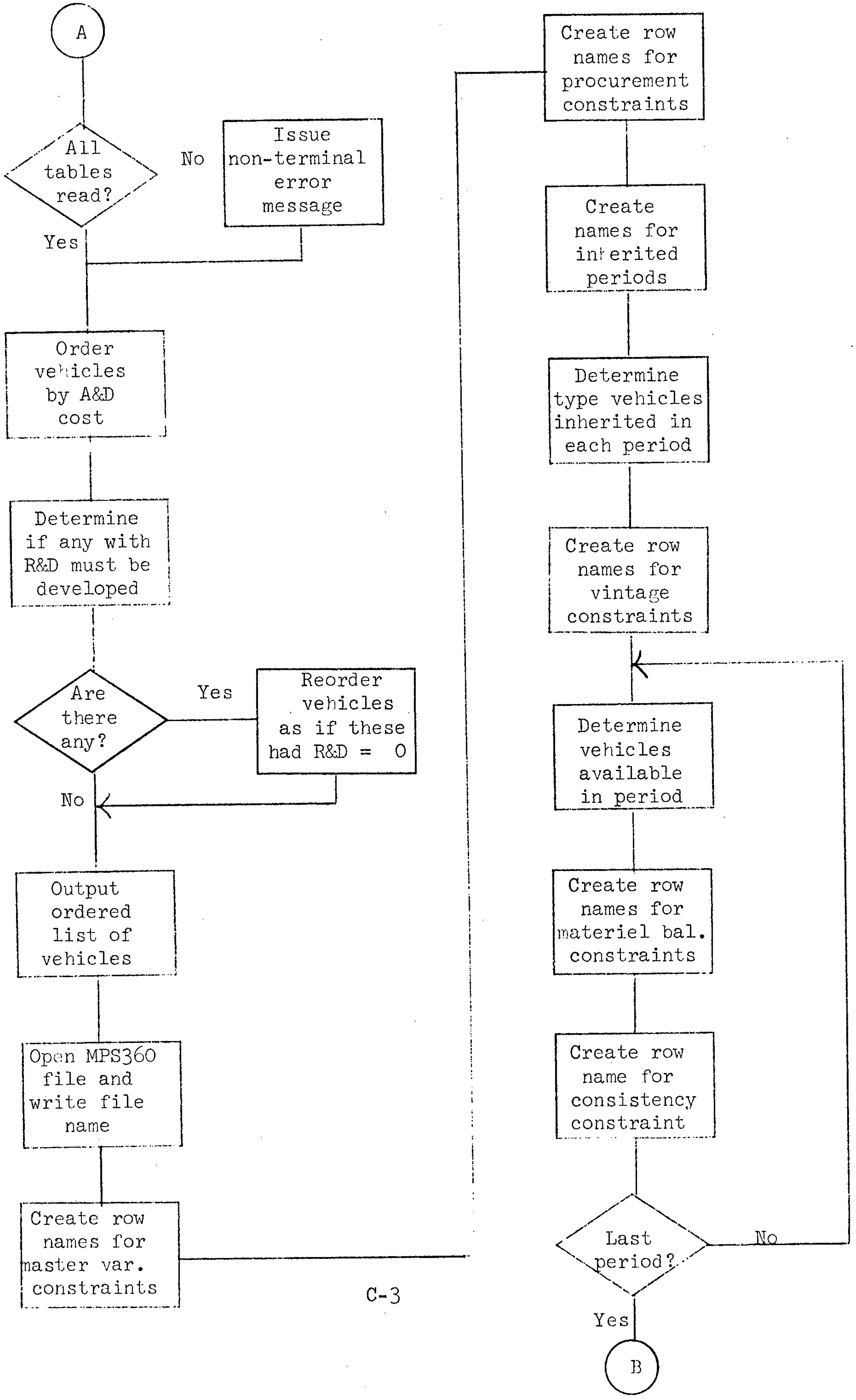
GENLCP............................ C-2

BBCAV2............................ C-12

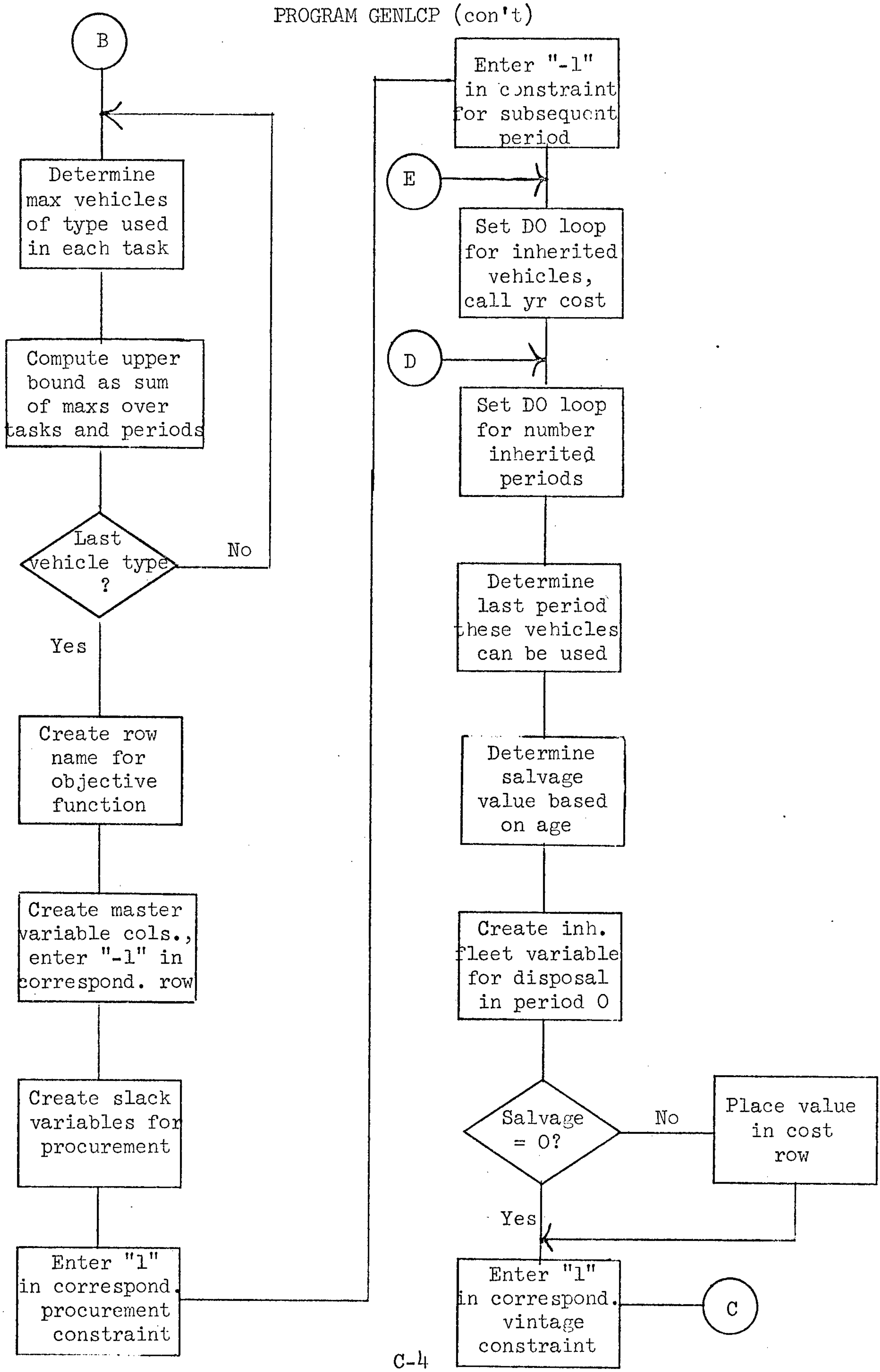
REPGEN............................ C-82

PROGRAM GENLCP

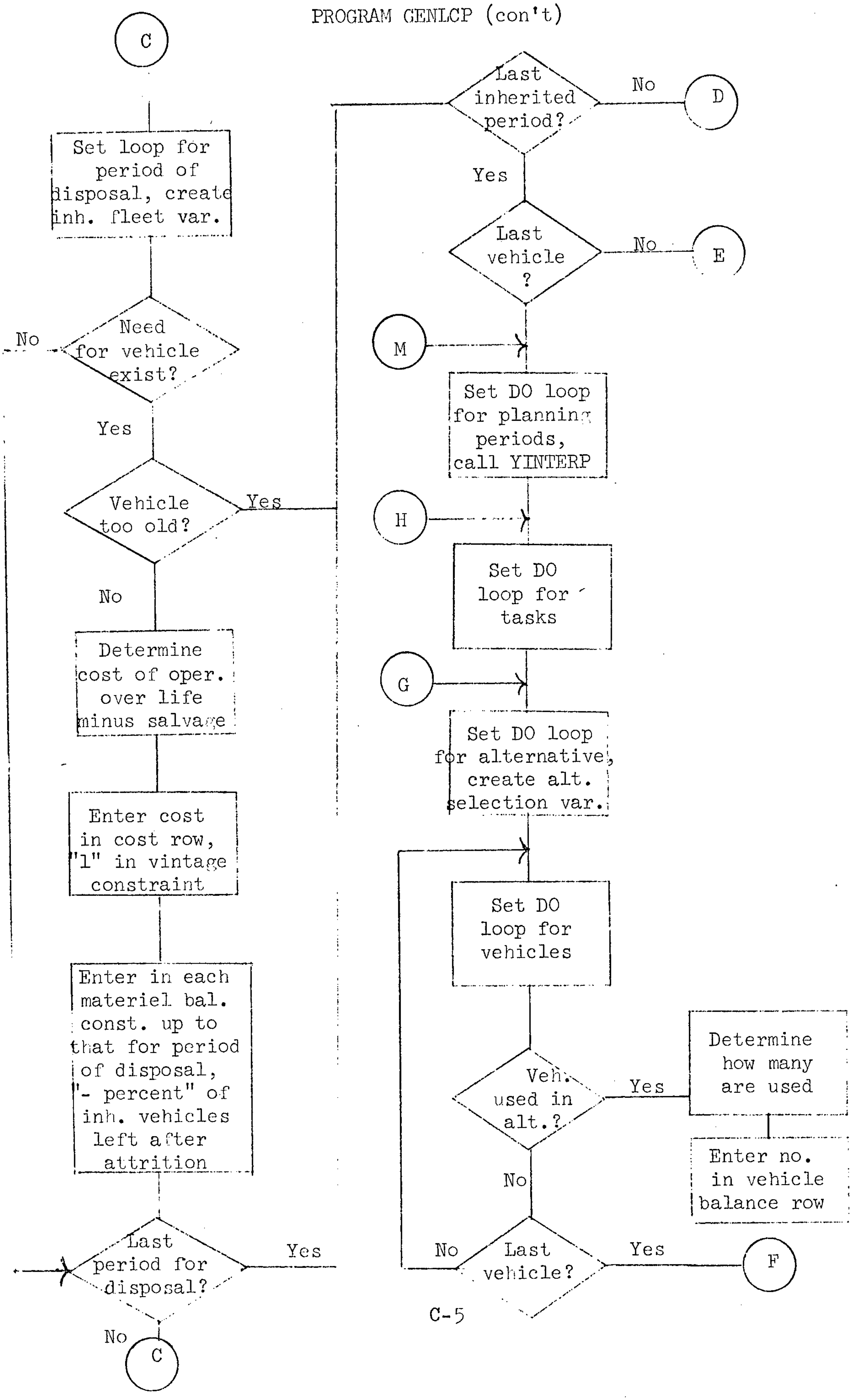



A
All tables read?
No
Issue non-terminal error message
Yes
Order vehicles by A&D cost
Determine if any with R&D must be developed
Are there any?
Yes
Reorder vehicles as if these had R&D = 0
No
Output ordered list of vehicles
Open MPS360 file and write file name
Create row names for master var. constraints
Create row names for procurement constraints
Create names for inherited periods
Determine type vehicles inherited in each period
Create row names for vintage constraints
Determine vehicles available in period
Create row names for materiel bal. constraints
Create row name for consistency constraint
Last period?
No
Yes
B

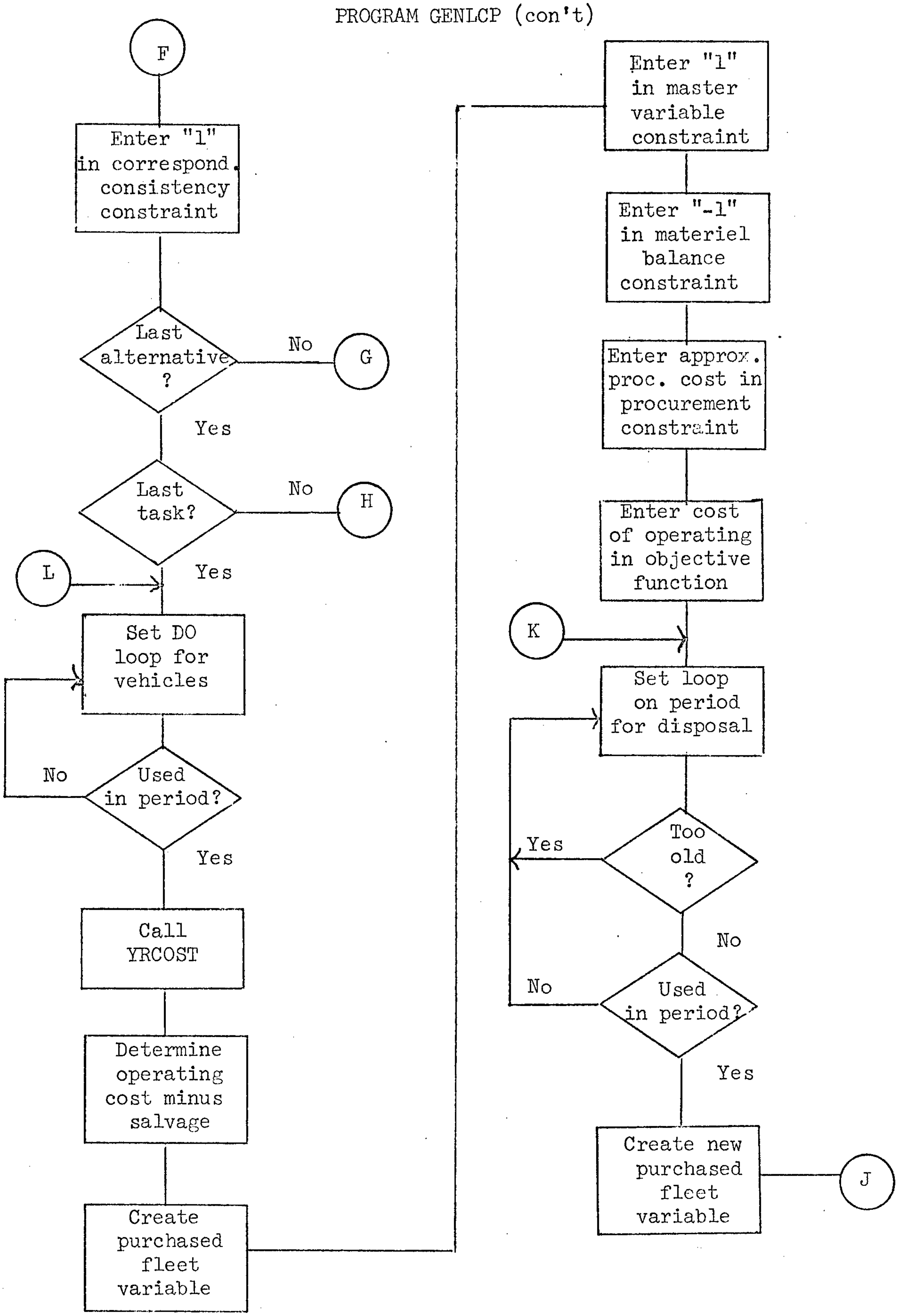
PROGRAM GENLCP (con't)


B
Determine max vehicles of type used in each task
Compute upper bound as sum of maxs over tasks and periods
Last vehicle type ?
No
Yes
Create row name for objective function
Create master variable cols., enter "-1" in correspond. row
Create slack variables for procurement
Enter "1" in correspond. procurement constraint
Enter "-1" in constraint for subsequent period
E
Set DO loop for inherited vehicles, call yr cost
D
Set DO loop for number inherited periods
Determine last period these vehicles can be used
Determine salvage value based on age
Create inh. fleet variable for disposal in period 0
Salvage = 0?
No
Place value in cost row
Yes
Enter "1" in correspond. vintage constraint
C

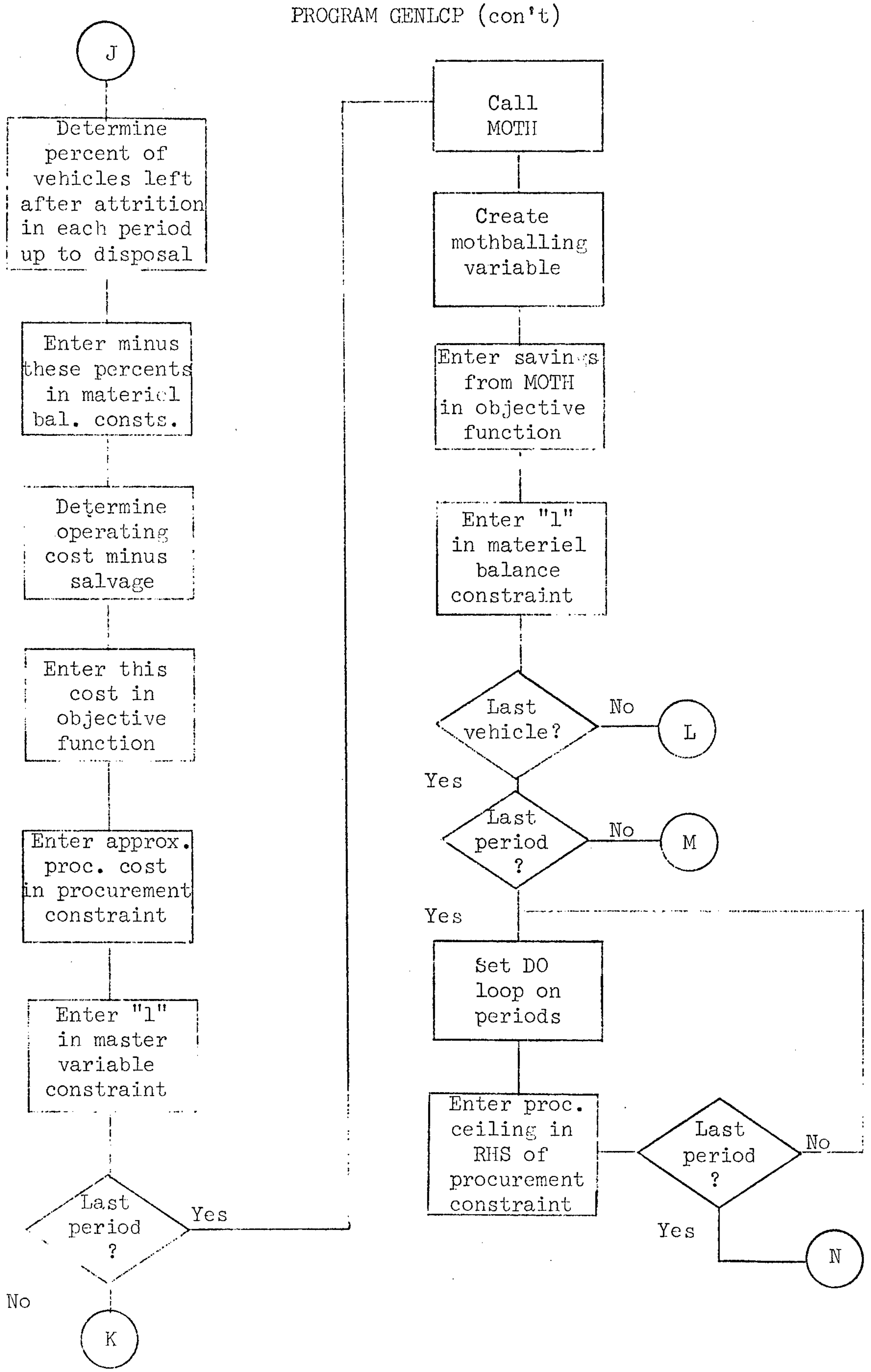
PROGRAM GENLCP (con't)


C
Set loop for period of disposal, create inh. fleet var.
Need for vehicle exist?
No
Yes
Vehicle too old?
Yes
No
Determine cost of oper. over life minus salvage
Enter cost in cost row, "1" in vintage constraint
Enter in each materiel bal. const. up to that for period of disposal, "- percent" of inh. vehicles left after attrition
Last period for disposal?
Yes
No
C
Last inherited period?
No
D
Yes
Last vehicle ?
No
E
M
Set DO loop for planning periods, call YINTERP
H
Set DO loop for tasks
G
Set DO loop for alternative, create alt. selection var.
Set DO loop for vehicles
Veh. used in alt.?
Yes
Determine how many are used
Enter no. in vehicle balance row
No
Last vehicle?
No
Yes
F

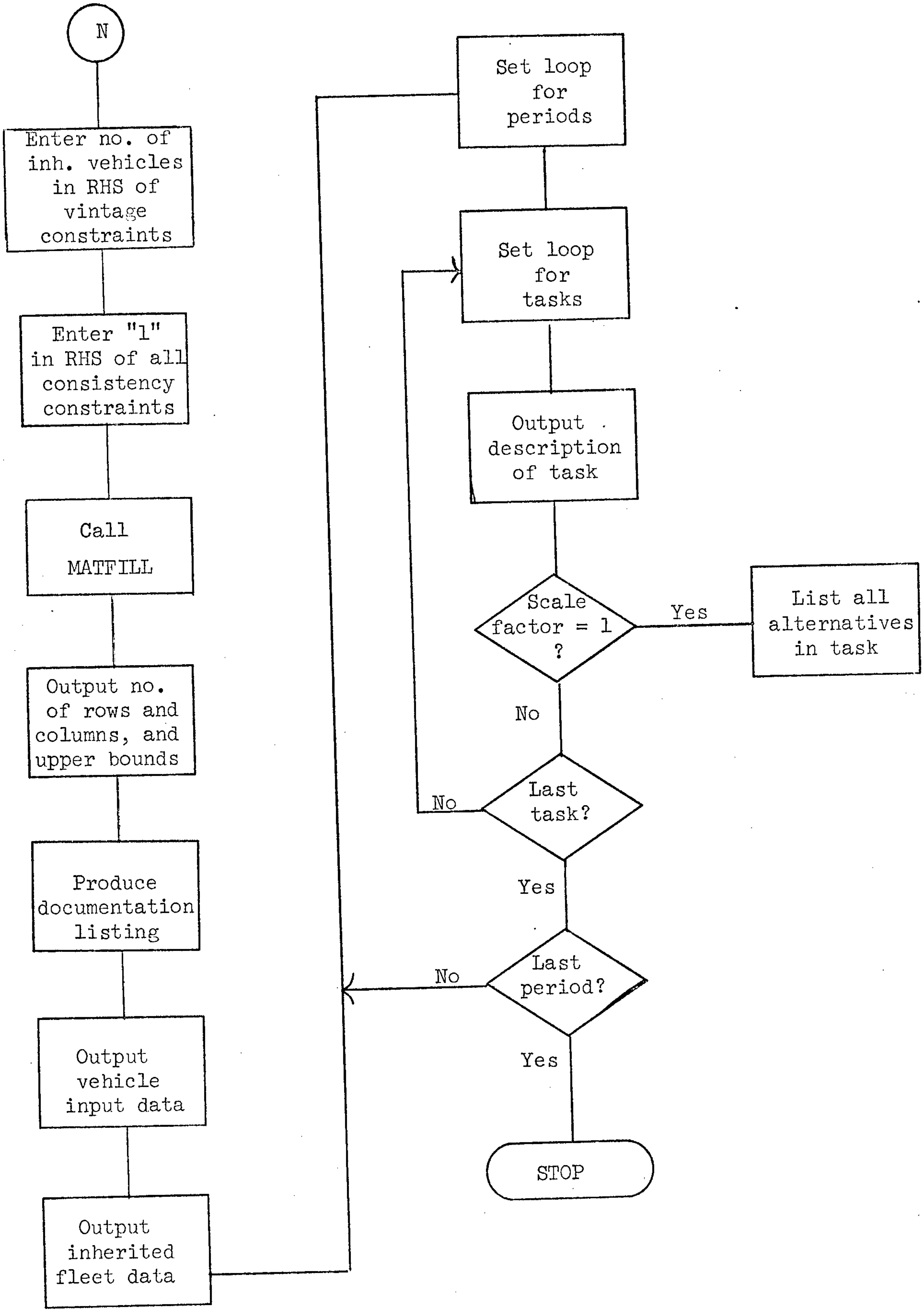
PROGRAM GENLCP (con't)


F
Enter "1" in correspond. consistency constraint
Last alternative ?
No
G
Yes
Last task?
No
H
L
Yes
Set DO loop for vehicles
No
Used in period?
Yes
Call YRCOST
Determine operating cost minus salvage
Create purchased fleet variable
Enter "1" in master variable constraint
Enter "-1" in materiel balance constraint
Enter approx. proc. cost in procurement constraint
Enter cost of operating in objective function
K
Set loop on period for disposal
Yes
Too old ?
No
No
Used in period?
Yes
Create new purchased fleet variable
J

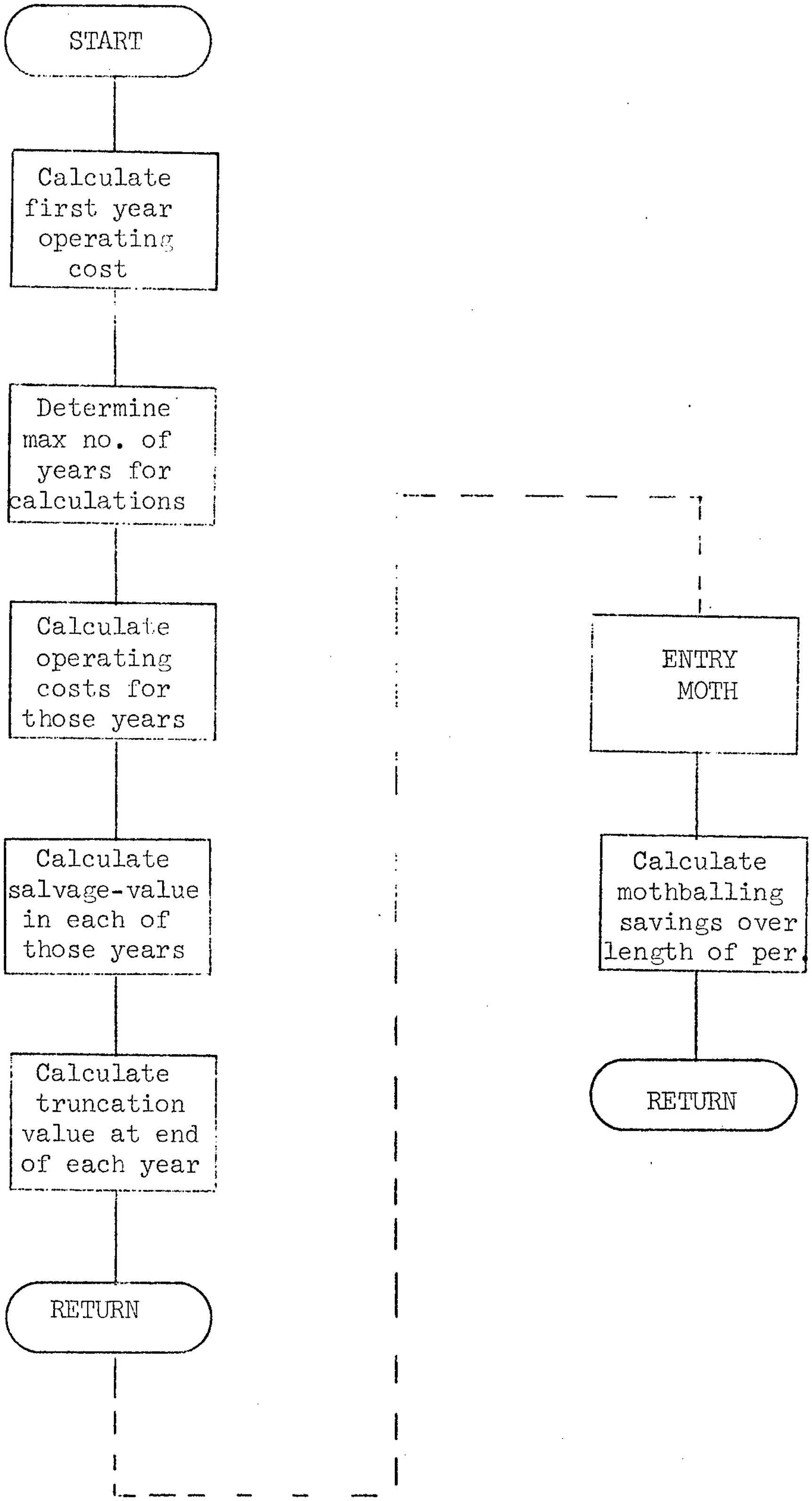
PROGRAM GENLCP (con't)


J
Determine percent of vehicles left after attrition in each period up to disposal
Enter minus these percents in materiel bal. consts.
Determine operating cost minus salvage
Enter this cost in objective function
Enter approx. proc. cost in procurement constraint
Enter "1" in master variable constraint
Last period ?
Yes
No
K
Call MOTH
Create mothballing variable
Enter savings from MOTH in objective function
Enter "1" in materiel balance constraint
Last vehicle?
No
L
Yes
Last period ?
No
M
Yes
Set DO loop on periods
Enter proc. ceiling in RHS of procurement constraint
Last period ?
No
Yes
N

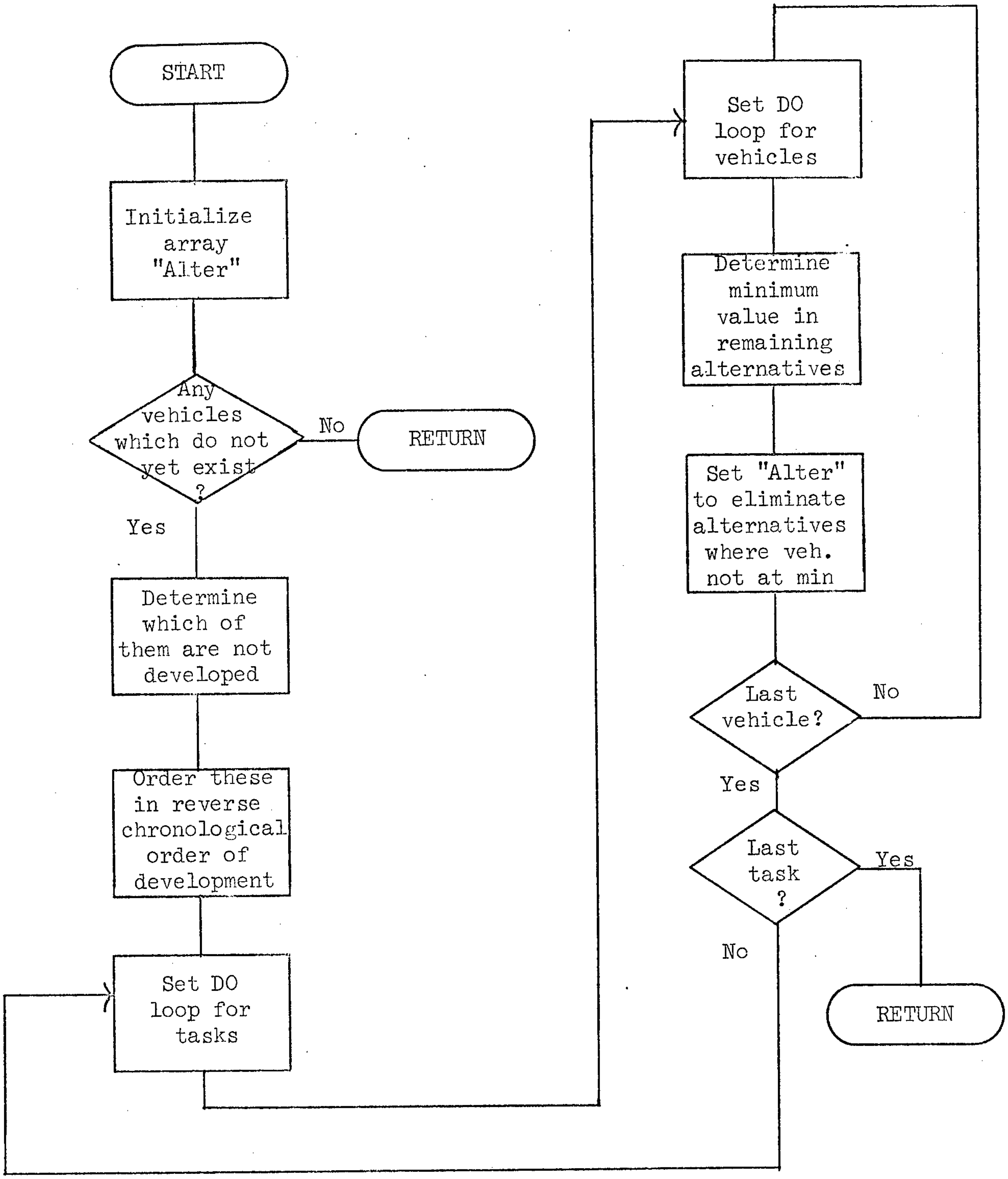
PROGRAM GENLCP


N
Enter no. of inh. vehicles in RHS of vintage constraints
Enter "1" in RHS of all consistency constraints
Call MATFILL
Output no. of rows and columns, and upper bounds
Produce documentation listing
Output vehicle input data
Output inherited fleet data
Set loop for periods
Set loop for tasks
Output description of task
Scale factor = 1 ?
Yes
List all alternatives in task
No
Last task?
No
Yes
Last period?
No
Yes
STOP

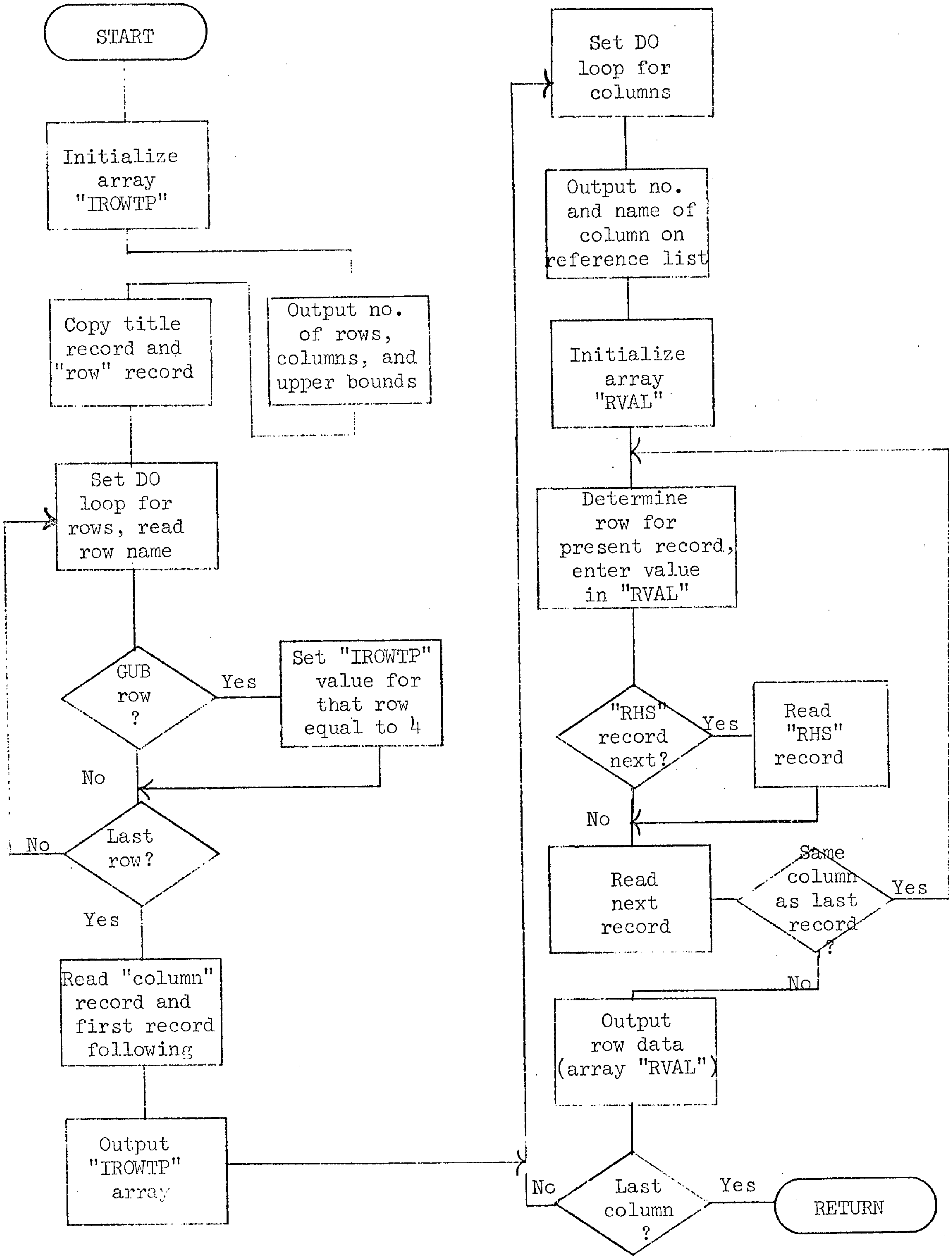
SUBROUTINE YRCOST


START
Calculate first year operating cost
Determine max no. of years for calculations
Calculate operating costs for those years
Calculate salvage-value in each of those years
Calculate truncation value at end of each year
RETURN
ENTRY MOTH
Calculate mothballing savings over length of per.
RETURN

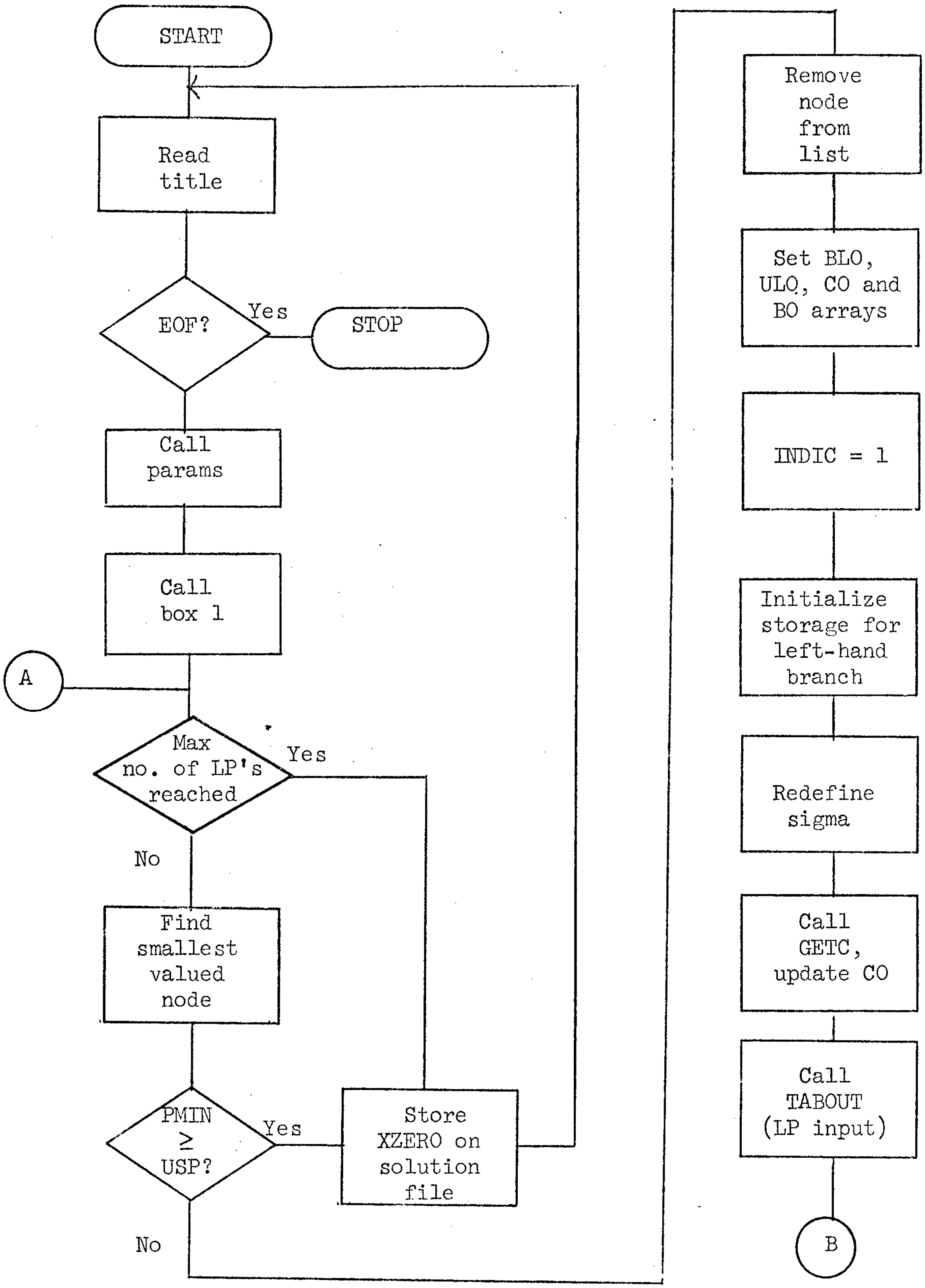
SUBROUTINE YINTERP


START
Initialize array "Alter"
Any vehicles which do not yet exist ?
No
RETURN
Yes
Determine which of them are not developed
Order these in reverse chronological order of development
Set DO loop for tasks
Set DO loop for vehicles
Determine minimum value in remaining alternatives
Set "Alter" to eliminate alternatives where veh. not at min
Last vehicle?
No
Yes
Last task ?
Yes
RETURN
No

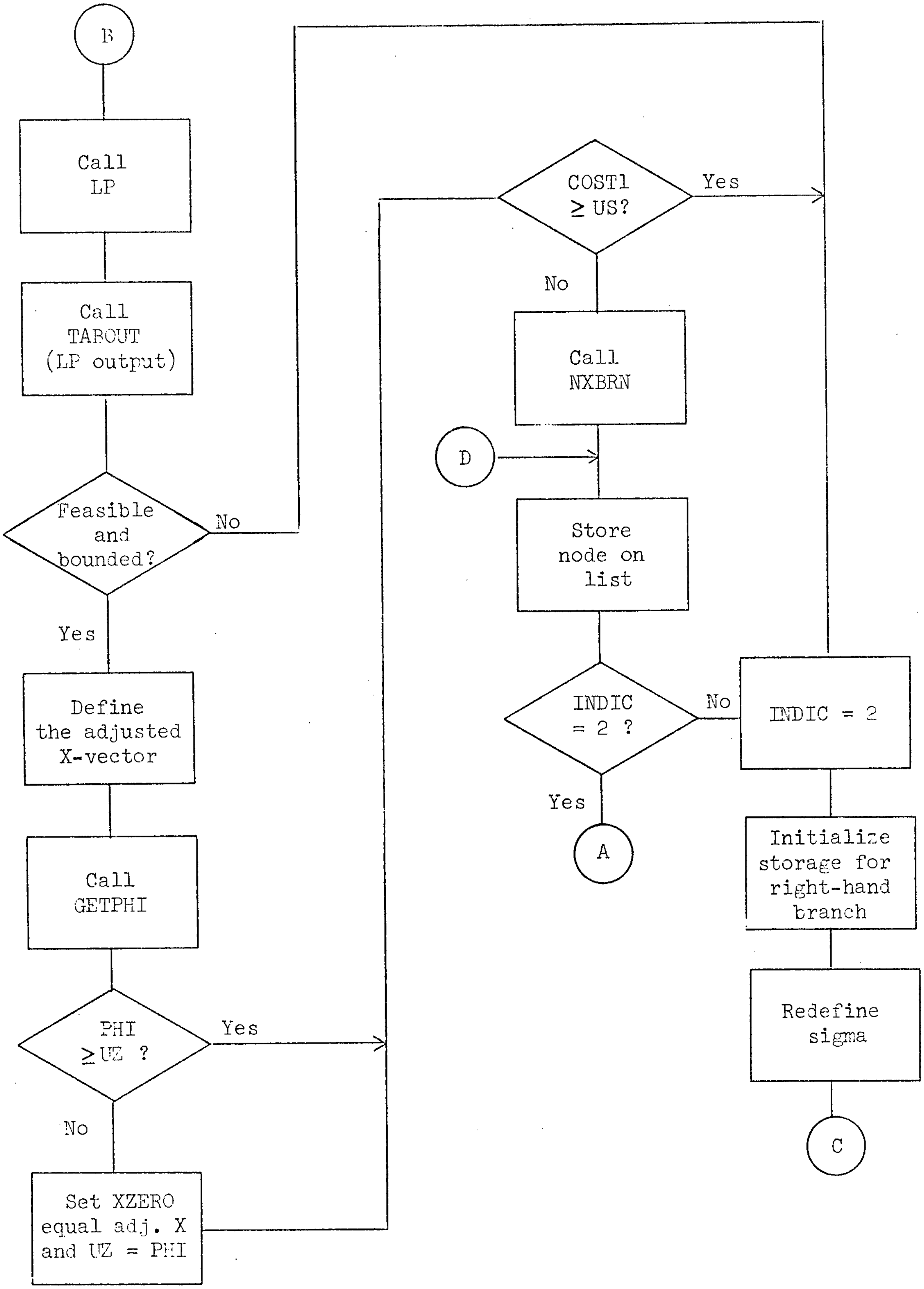
SUBROUTINE MATFILL


START
Initialize array "IROWTP"
Copy title record and "row" record
Output no. of rows, columns, and upper bounds
Set DO loop for rows, read row name
GUB row ?
Yes
Set "IROWTP" value for that row equal to 4
No
Last row?
No
Yes
Read "column" record and first record following
Output "IROWTP" array
Set DO loop for columns
Output no. and name of column on reference list
Initialize array "RVAL"
Determine row for present record, enter value in "RVAL"
"RHS" record next?
Yes
Read "RHS" record
No
Read next record
Same column as last record ?
Yes
No
Output row data (array "RVAL")
Last column ?
No
Yes
RETURN

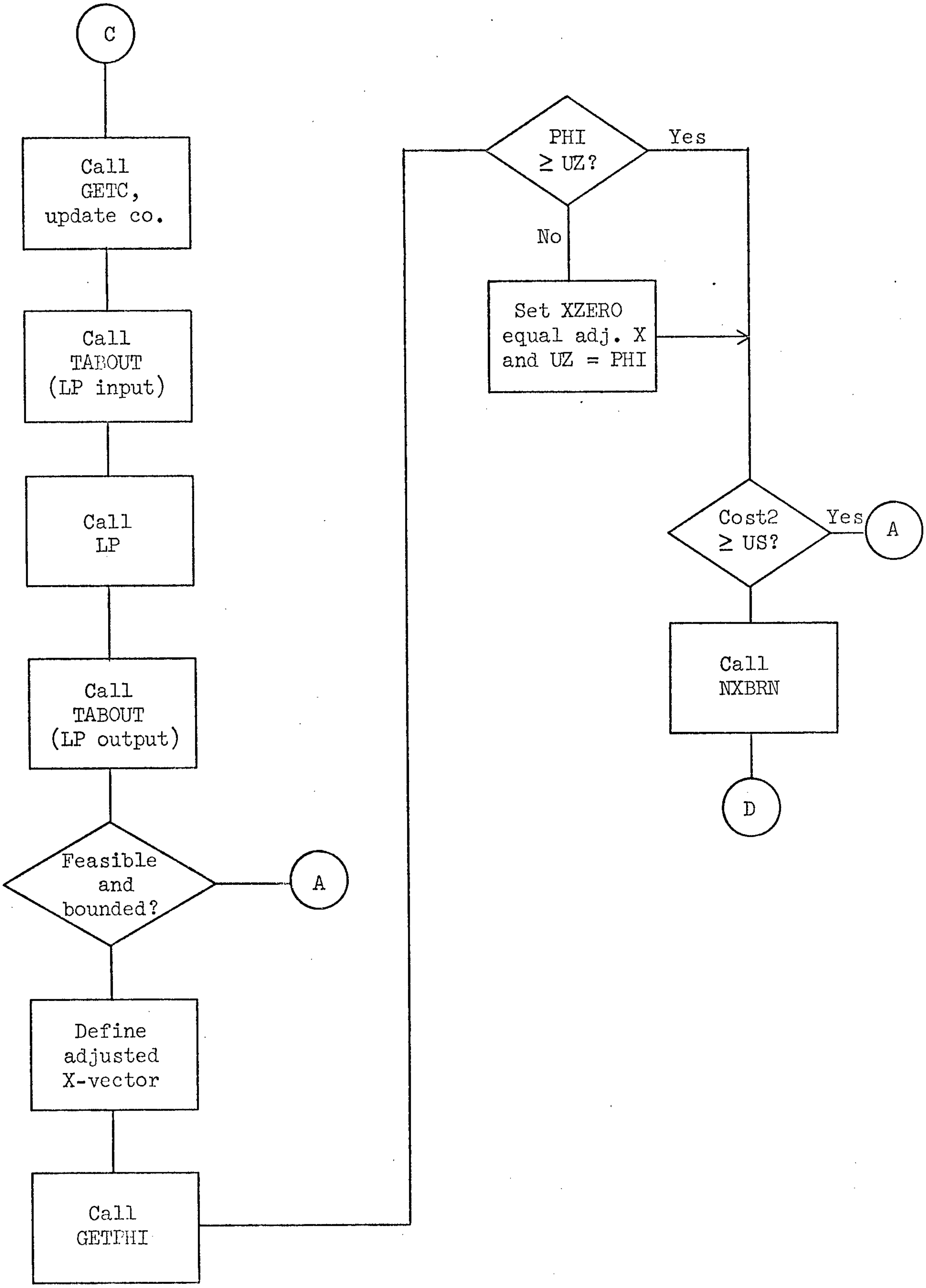
PROGRAM BBCAV2



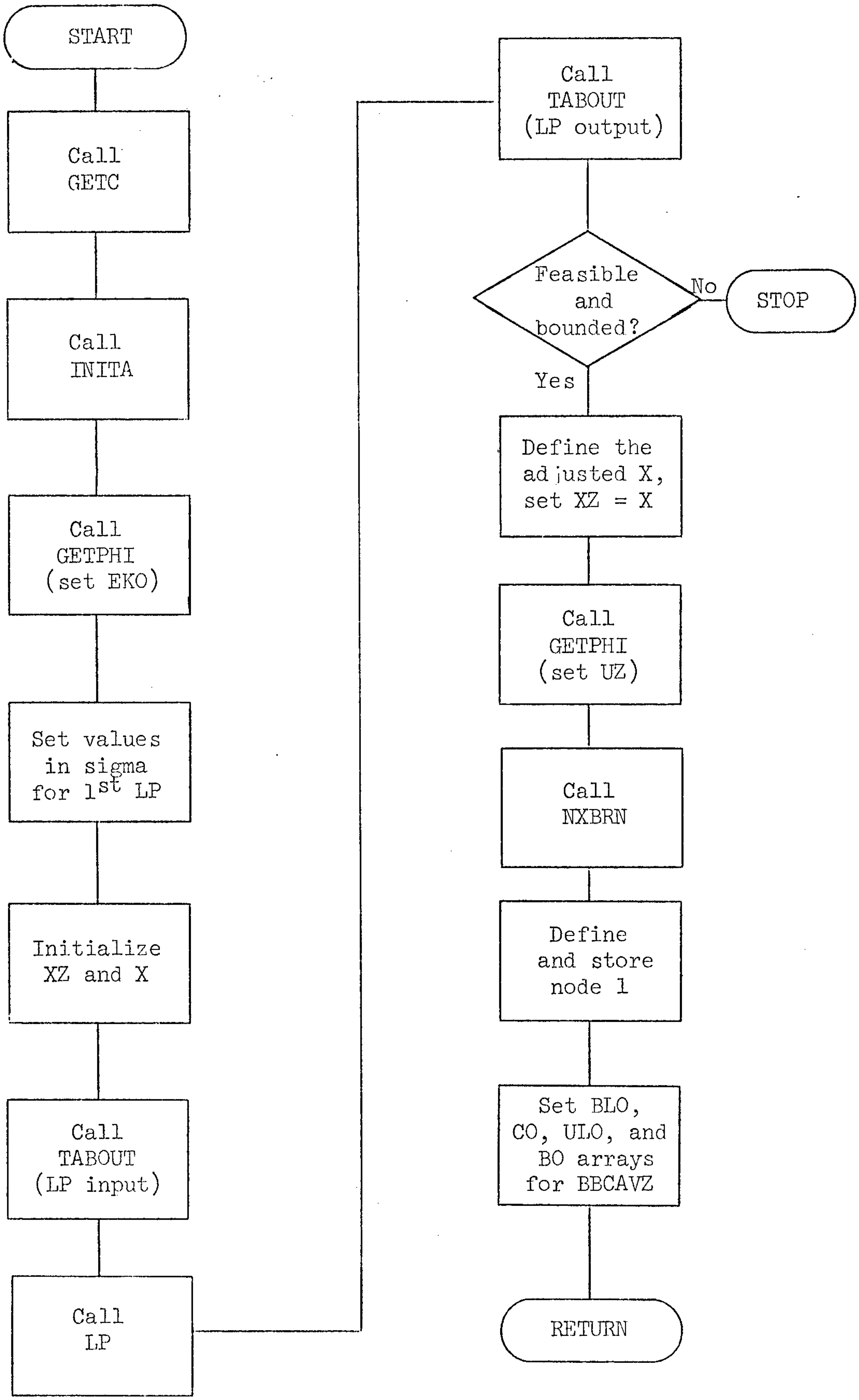
PROGRAM BBCAV2 (con't)



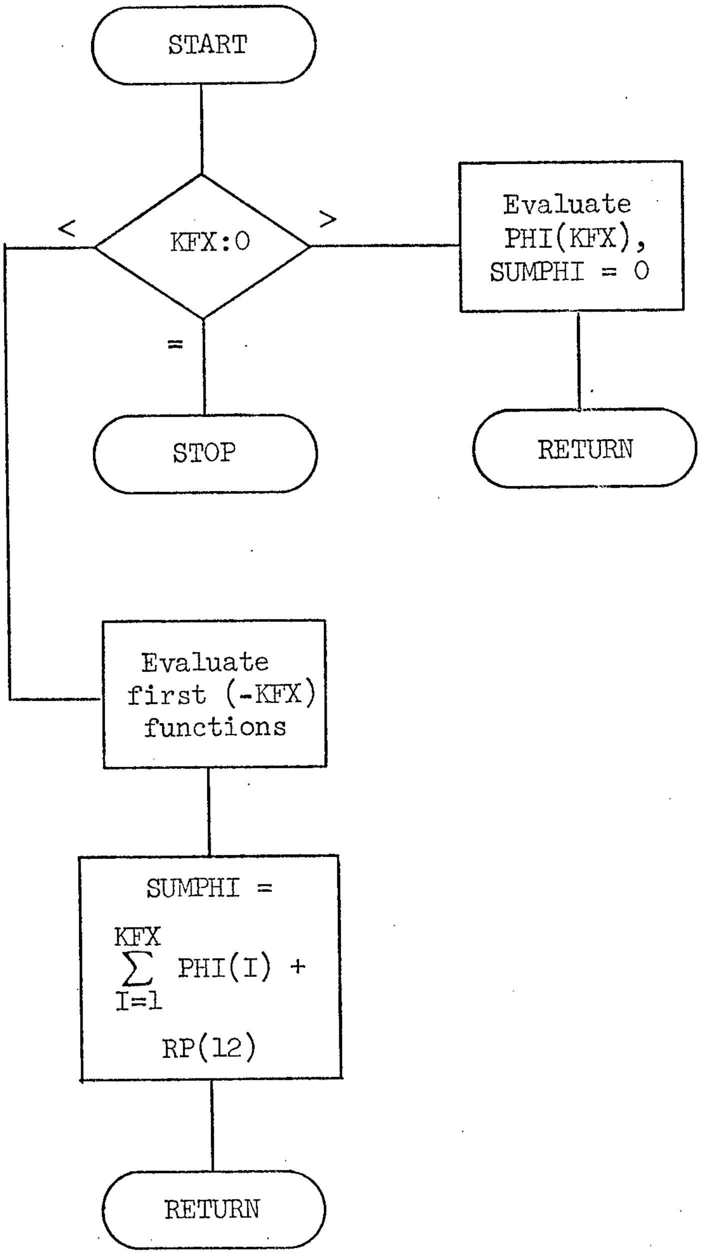
PROGRAM BBCAV2 (con't)



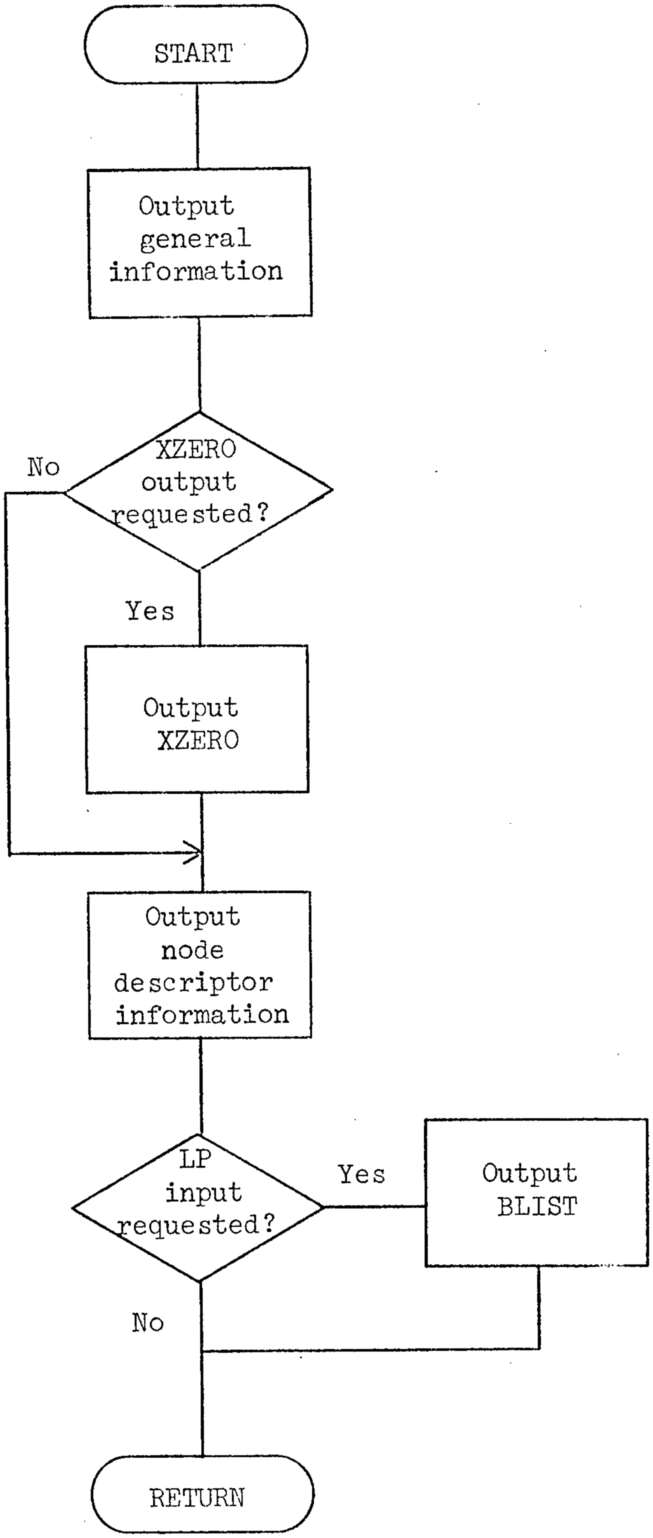
SUBROUTINE BOX1



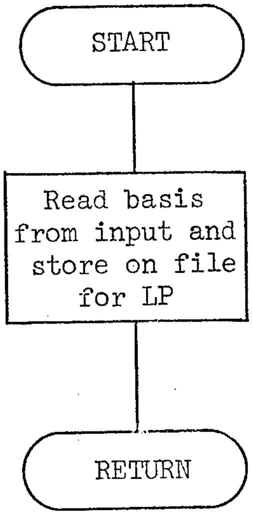
SUBROUTINE GETPHI


START
KFX:0
<
>
=
Evaluate PHI(KFX), SUMPHI = 0
STOP
RETURN
Evaluate first (-KFX) functions
SUMPHI = $\sum_{I=1}^{KFX}$ PHI(I) + RP(12)
RETURN

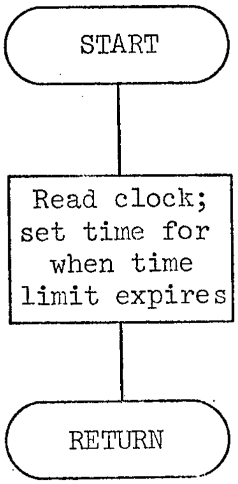
SUBROUTINE TABOUT



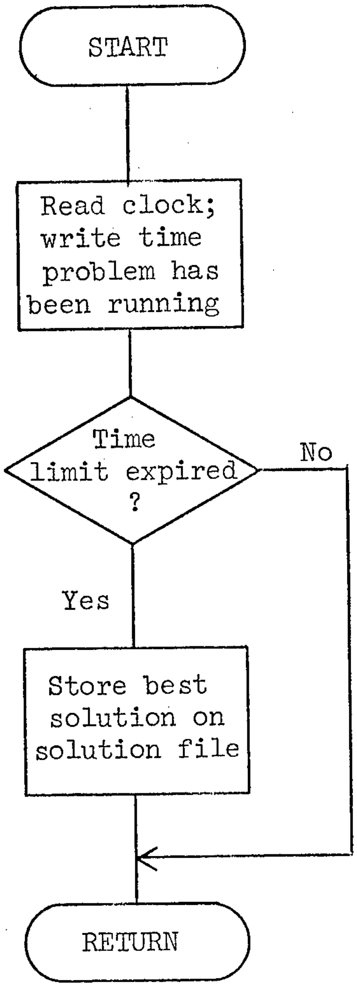
SUBROUTINE READIN



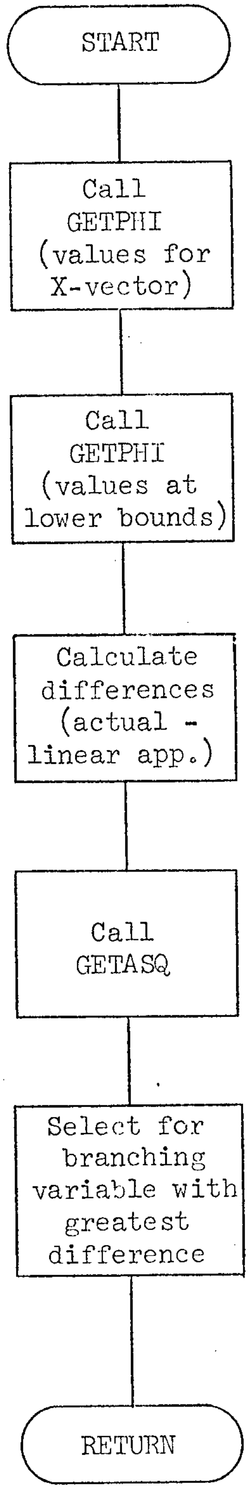
SUBROUTINE SET



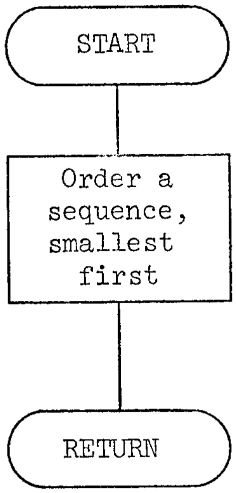
SUBROUTINE TIMEC


START
Read clock; write time problem has been running
Time limit expired ?
No
Yes
Store best solution on solution file
RETURN

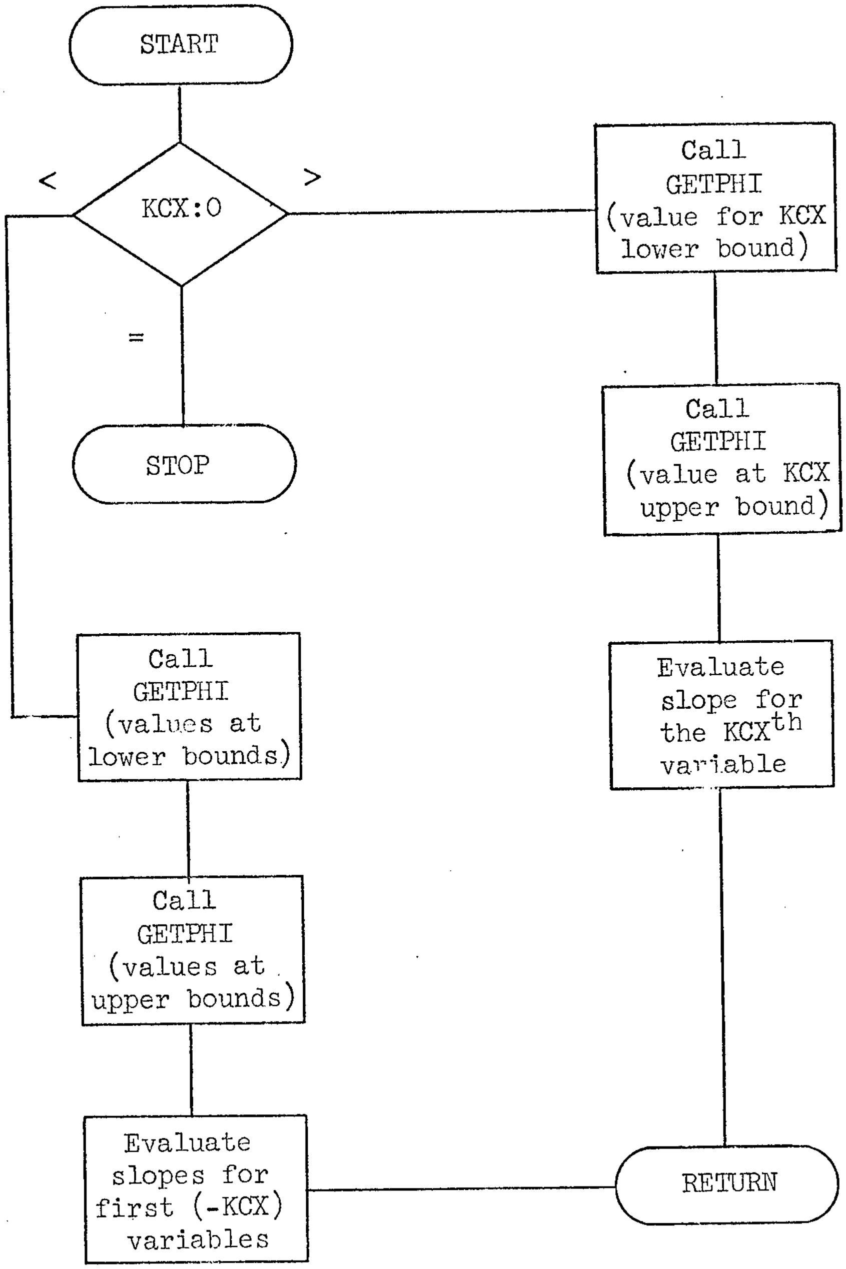
SUBROUTINE NXBRN


START
Call GETPHI (values for X-vector)
Call GETPHI (values at lower bounds)
Calculate differences (actual - linear app.)
Call GETASQ
Select for branching variable with greatest difference
RETURN

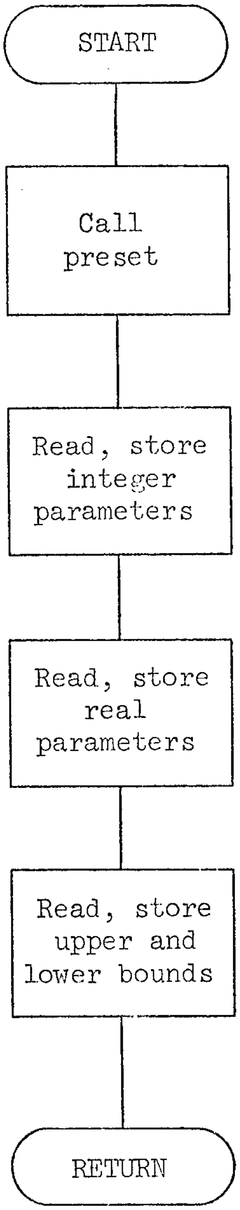
SUBROUTINE GETASQ



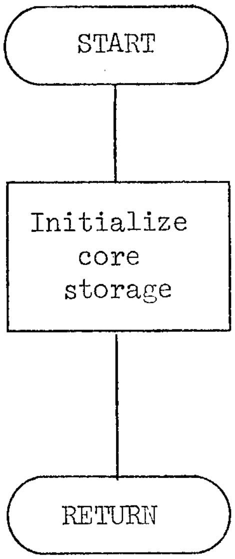
SUBROUTINE GETC


START
KCX:0
<
>
=
STOP
Call GETPHI (value for KCX lower bound)
Call GETPHI (value at KCX upper bound)
Evaluate slope for the KCX^th variable
Call GETPHI (values at lower bounds)
Call GETPHI (values at upper bounds)
Evaluate slopes for first (-KCX) variables
RETURN

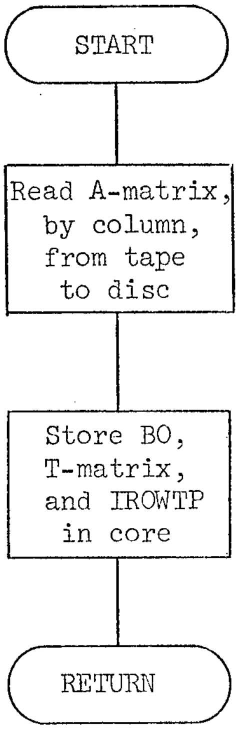
SUBROUTINE PARAMS


START
Call preset
Read, store integer parameters
Read, store real parameters
Read, store upper and lower bounds
RETURN

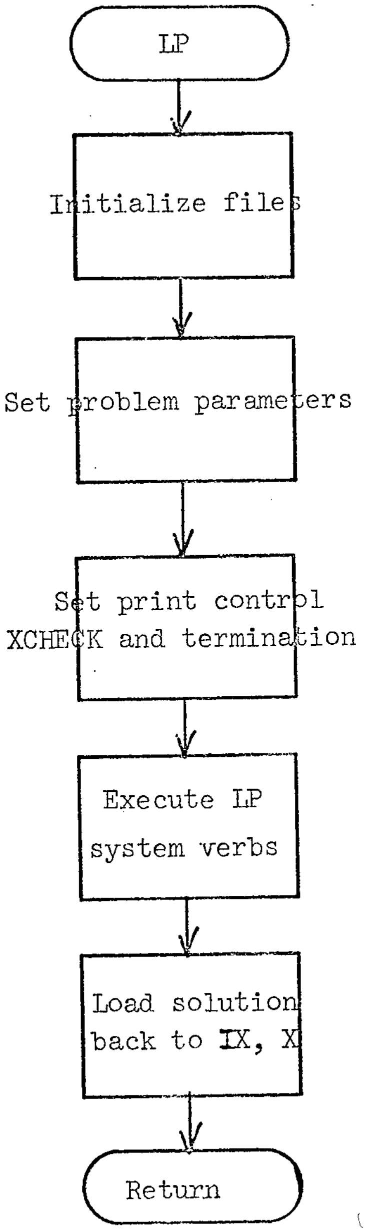
SUBROUTINE PRESET



SUBROUTINE INITA

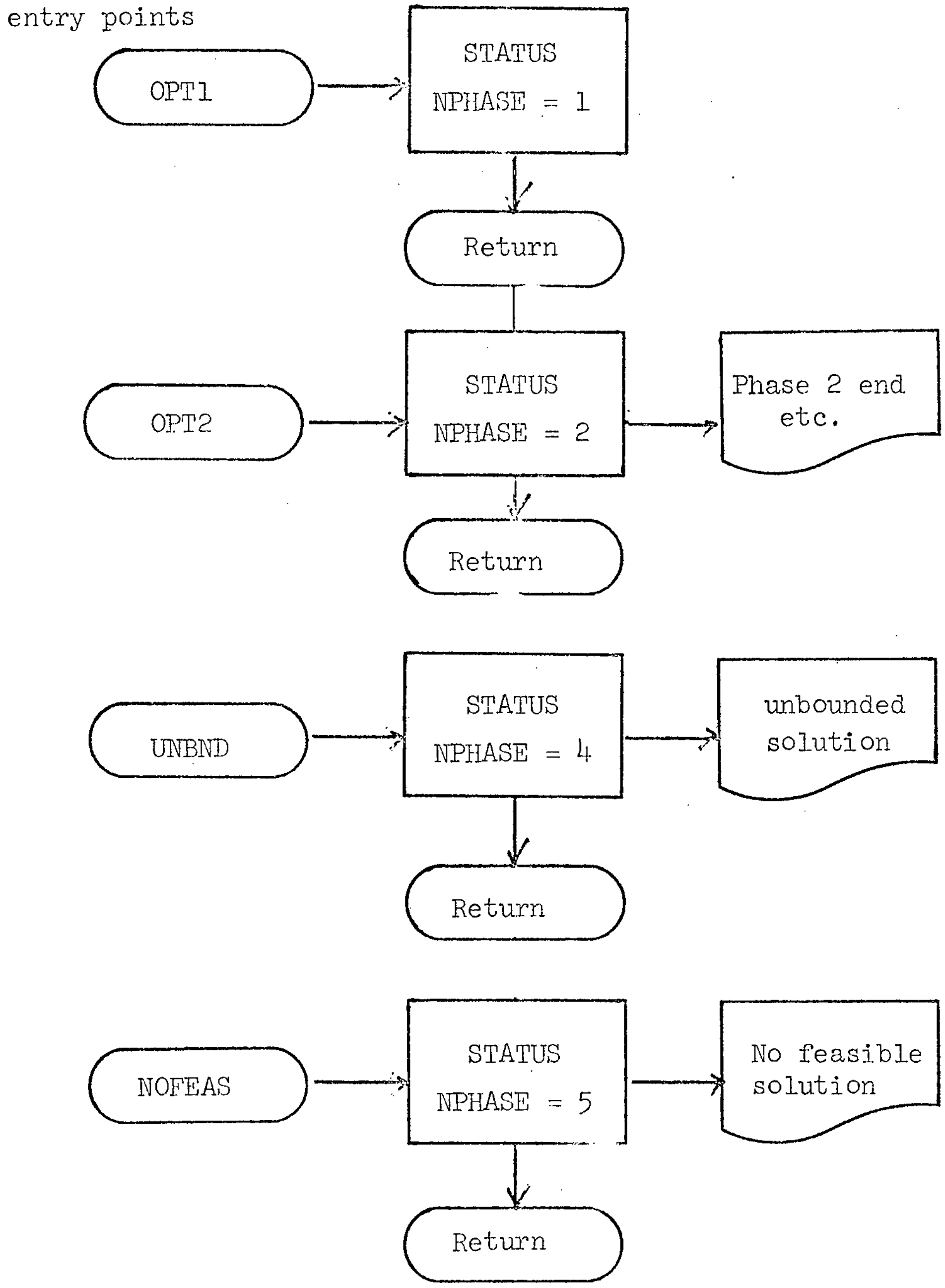

START
Read A-matrix, by column, from tape to disc
Store BO, T-matrix, and IROWTP in core
RETURN

Subroutine LP

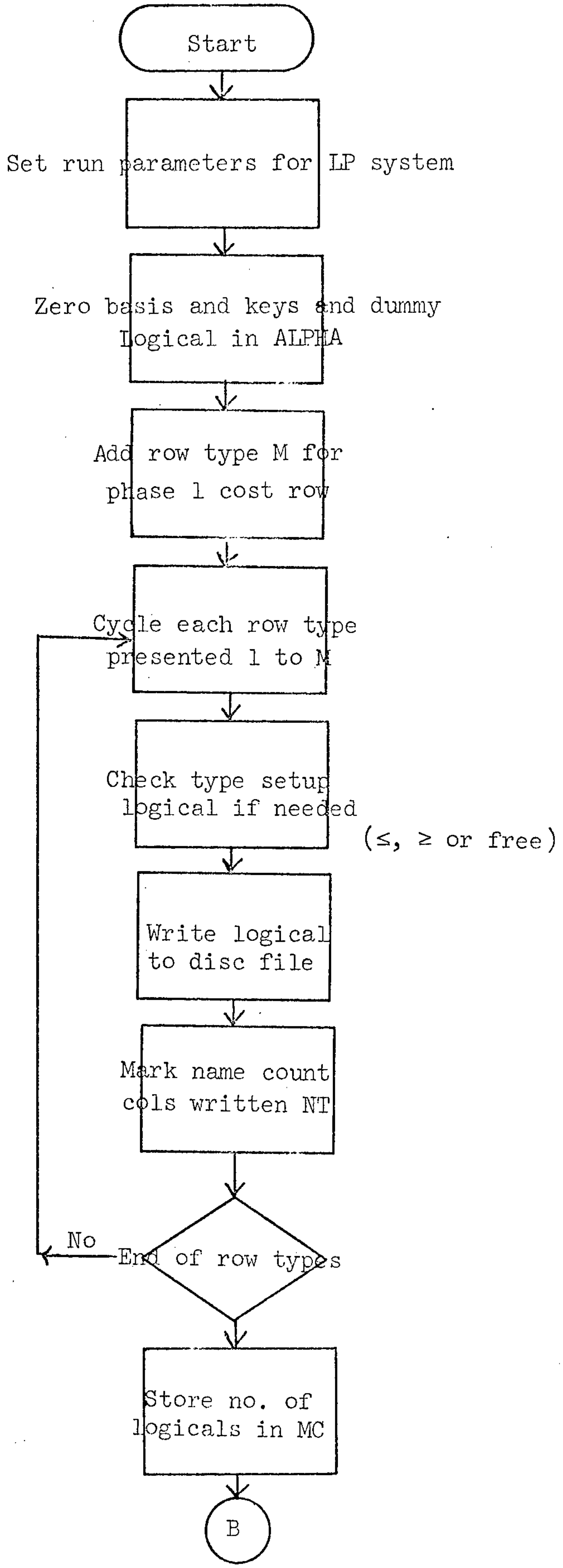


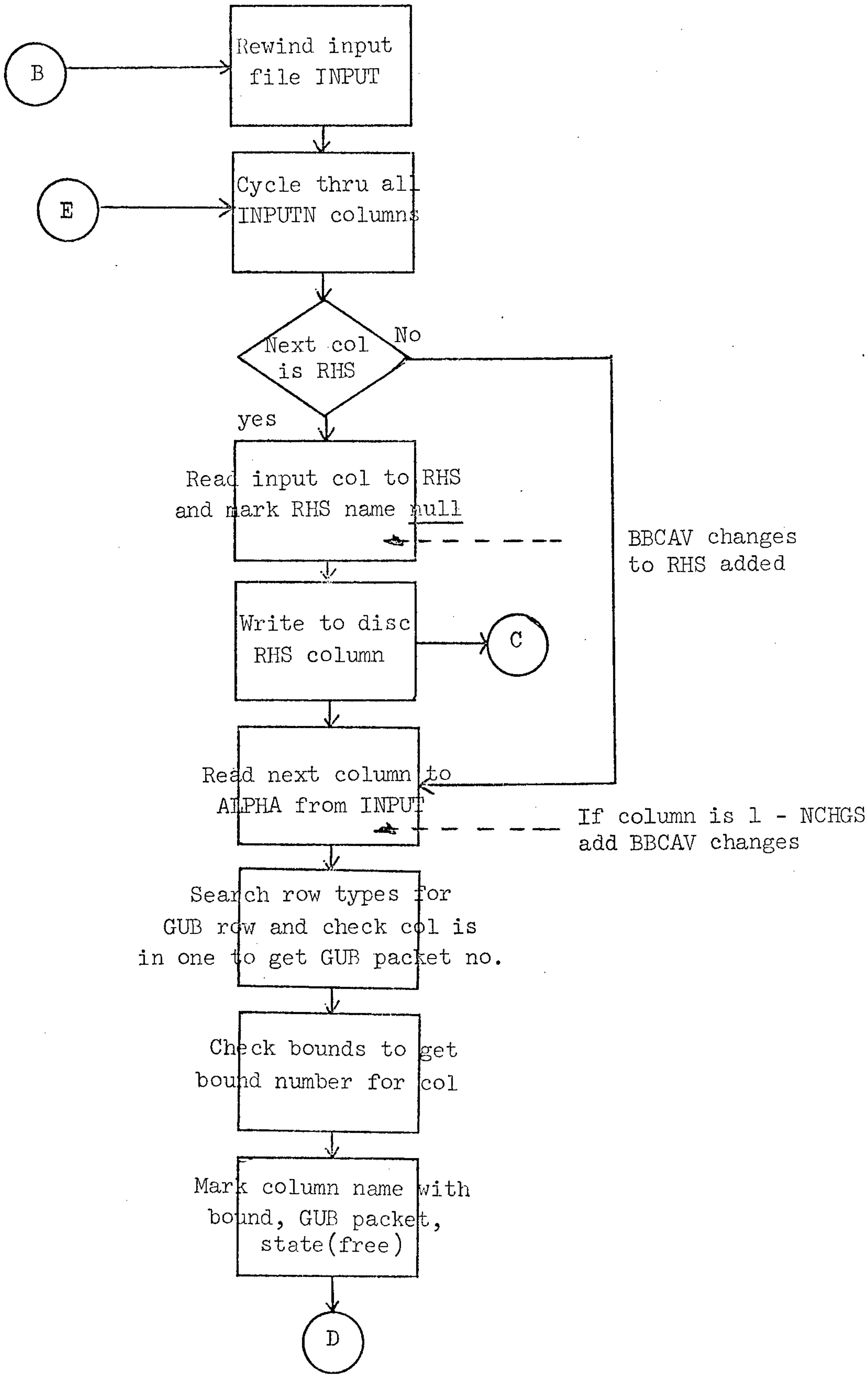
Subroutine EXITS

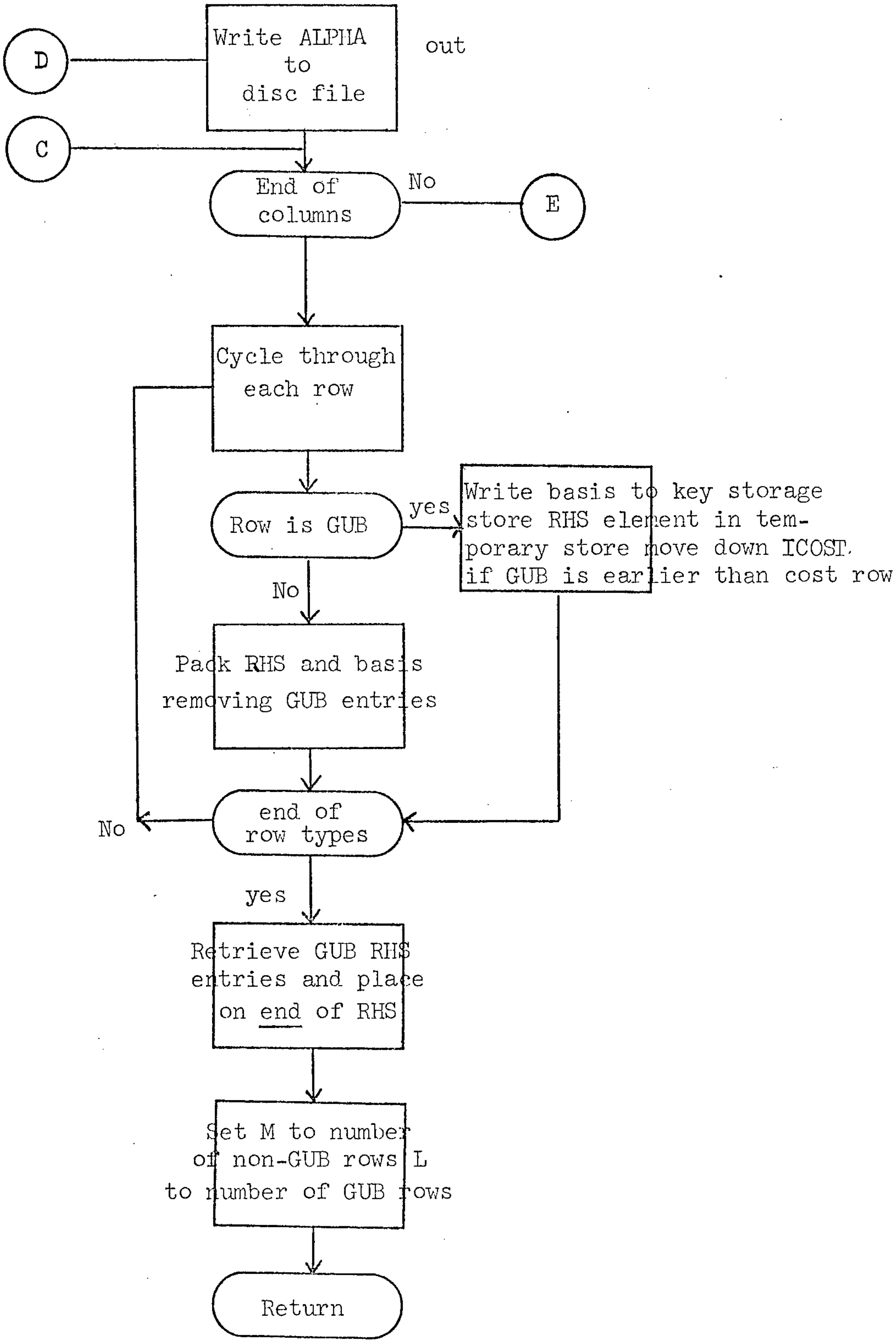
User changable EXITS program called after all control points in PRIMAL.



Subroutinc SETUP

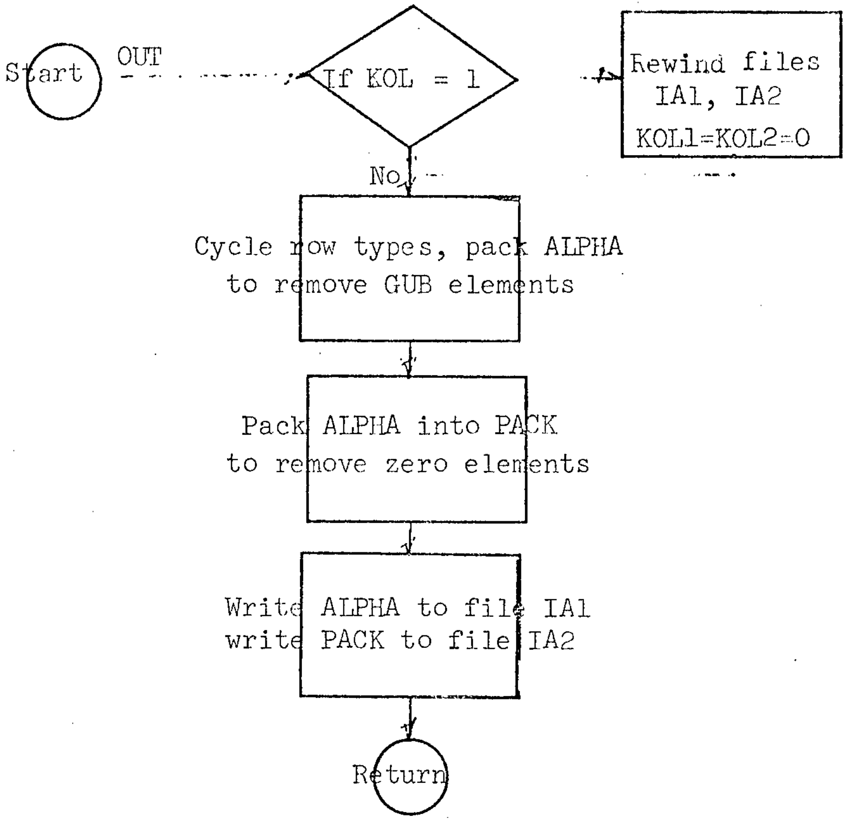

Start
Set run parameters for LP system
Zero basis and keys and dummy logical in ALPHA
Add row type M for phase 1 cost row
Cycle each row type presented 1 to M
Check type setup logical if needed
(≤, ≥ or free)
Write logical to disc file
Mark name count cols written NT
End of row types
No
Store no. of logicals in MC
B

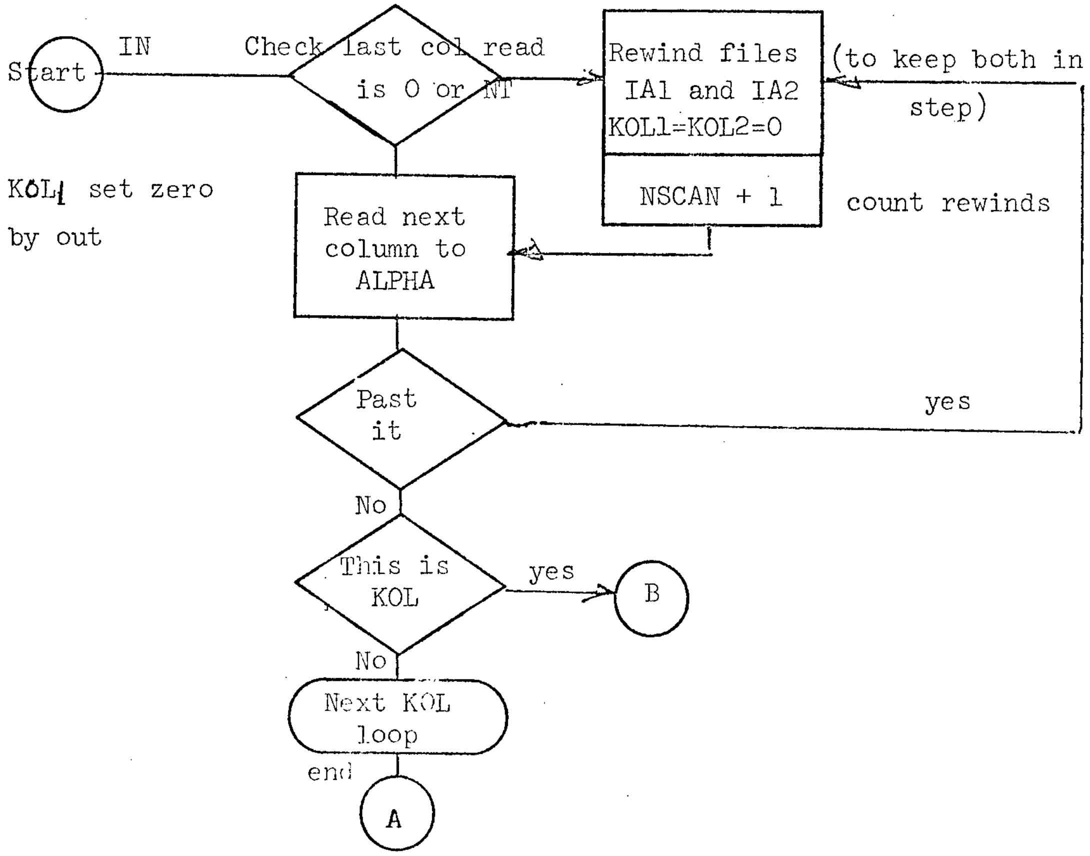

B
Rewind input file INPUT
E
Cycle thru all INPUTN columns
Next col is RHS
No
yes
Read input col to RHS and mark RHS name null
BBCAV changes to RHS added
Write to disc RHS column
C
Read next column to ALPHA from INPUT
If column is 1 - NCHGS add BBCAV changes
Search row types for GUB row and check col is in one to get GUB packet no.
Check bounds to get bound number for col
Mark column name with bound, GUB packet, state(free)
D

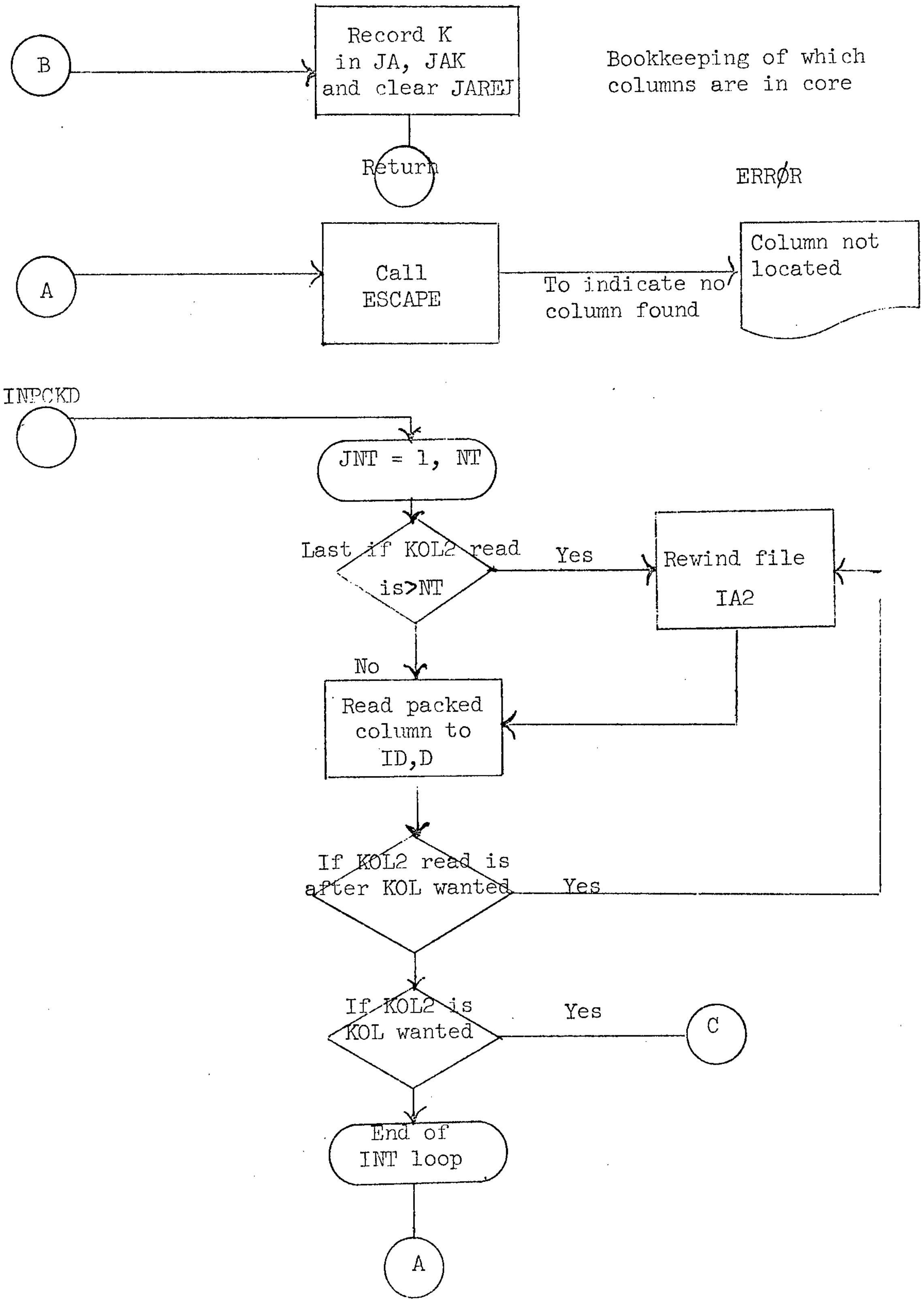

D
Write ALPHA to disc file
out
C
End of columns
No
E
Cycle through each row
Row is GUB
yes
Write basis to key storage store RHS element in temporary store move down ICOST. if GUB is earlier than cost row
No
Pack RHS and basis removing GUB entries
end of row types
No
yes
Retrieve GUB RHS entries and place on end of RHS
Set M to number of non-GUB rows L to number of GUB rows
Return

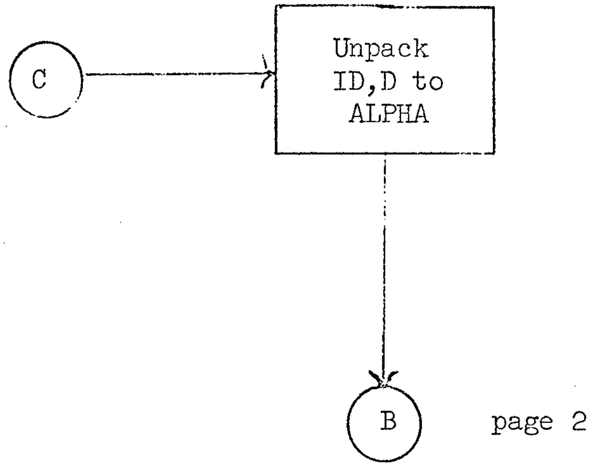
Subroutine IO

IO 1.





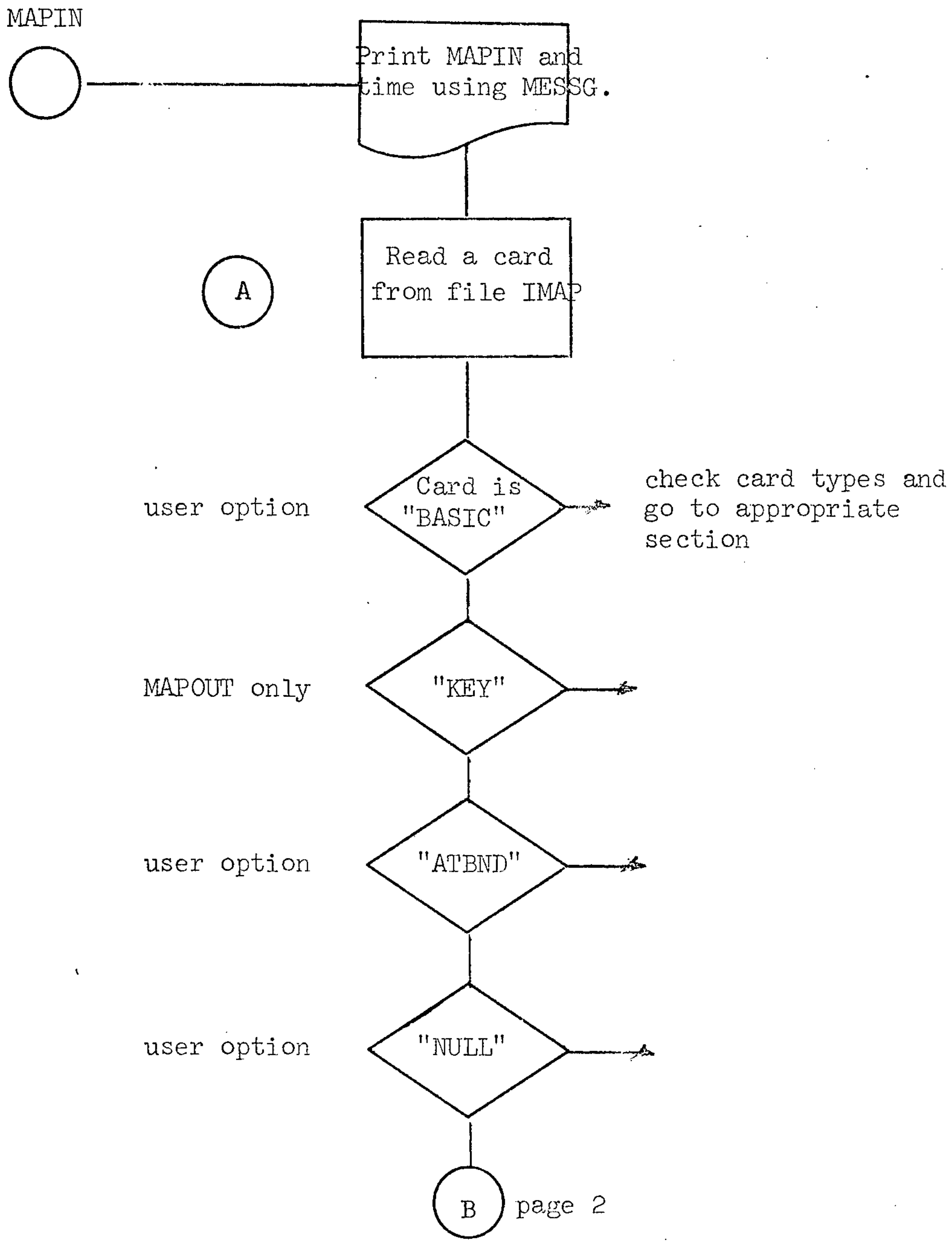

B
Record K in JA, JAK and clear JAREJ
Bookkeeping of which columns are in core
Return
ERRØR
A
Call ESCAPE
To indicate no column found
Column not located
INPCKD
JNT = 1, NT
Last if KOL2 read is>NT
Yes
Rewind file IA2
No
Read packed column to ID,D
If KOL2 read is after KOL wanted
Yes
If KOL2 is KOL wanted
Yes
C
End of INT loop
A

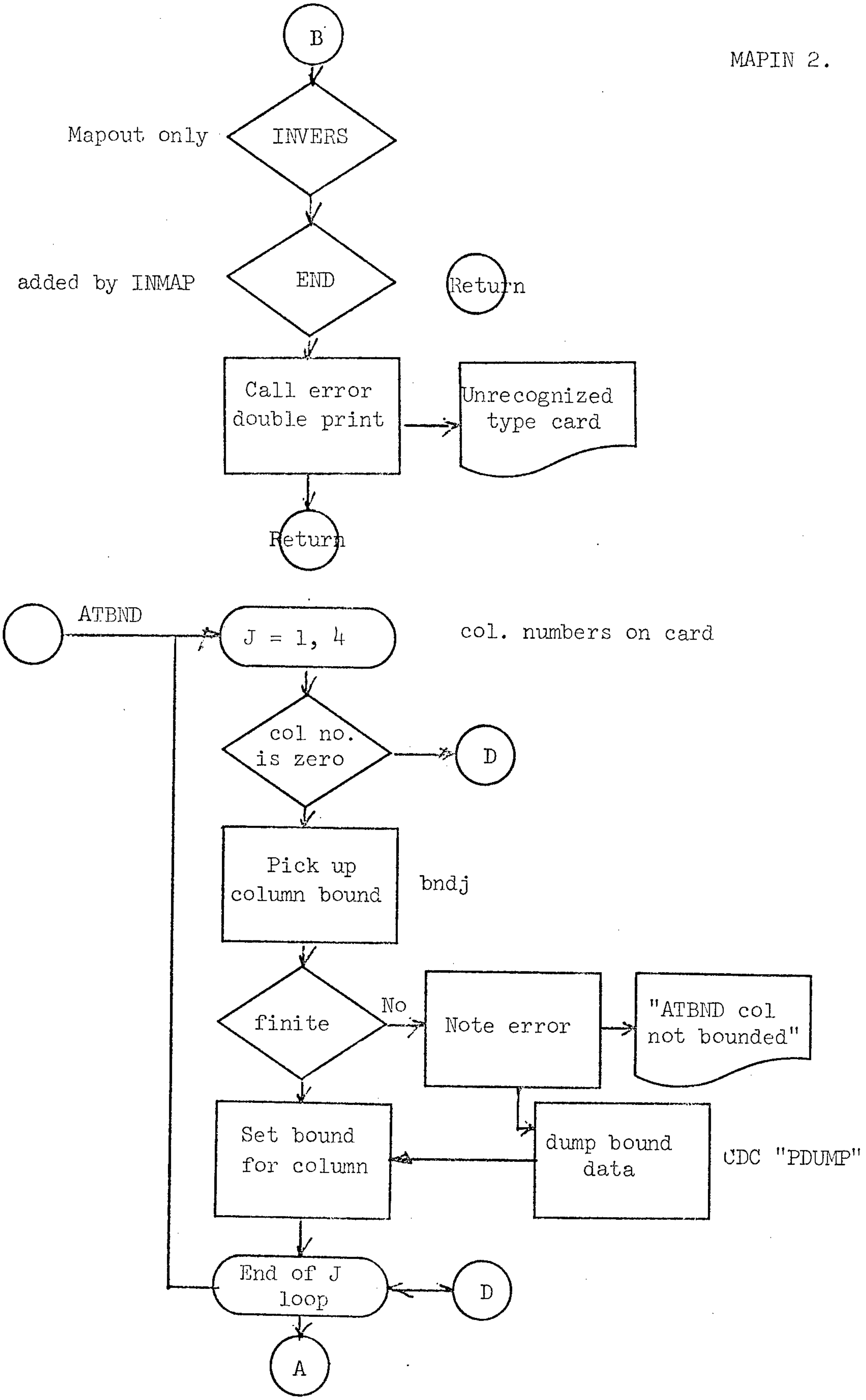

C
Unpack ID,D to ALPHA
B
page 2

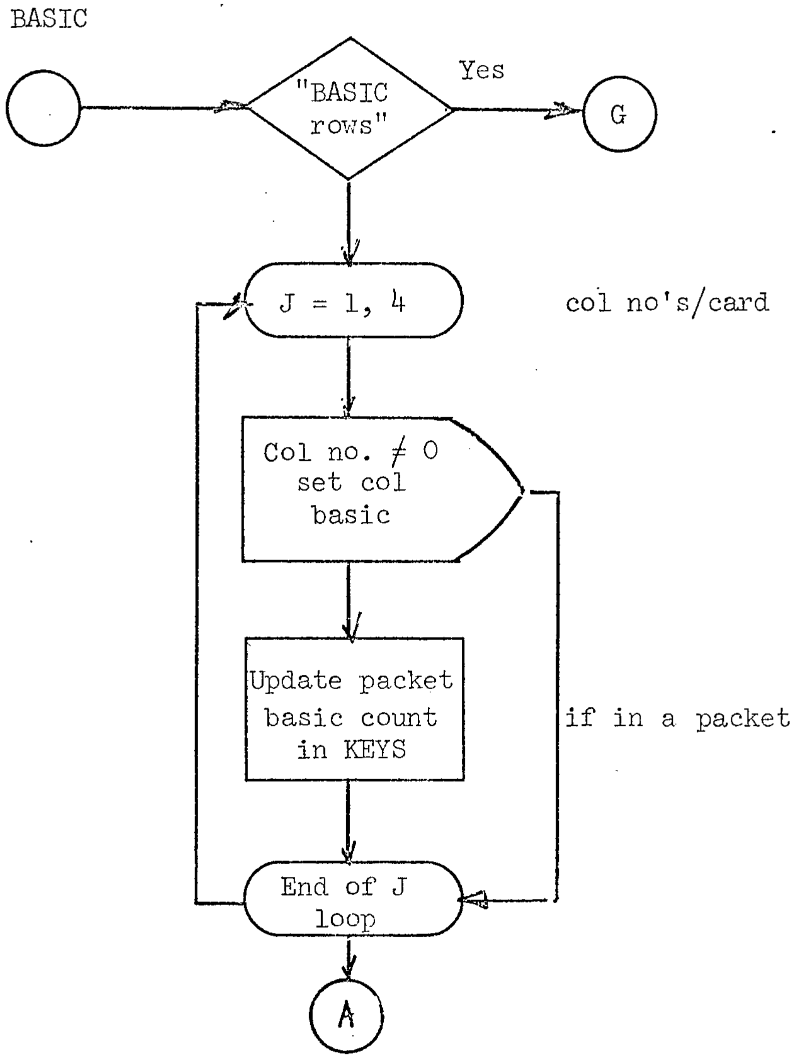
Subroutine MAPIN
Two entry points

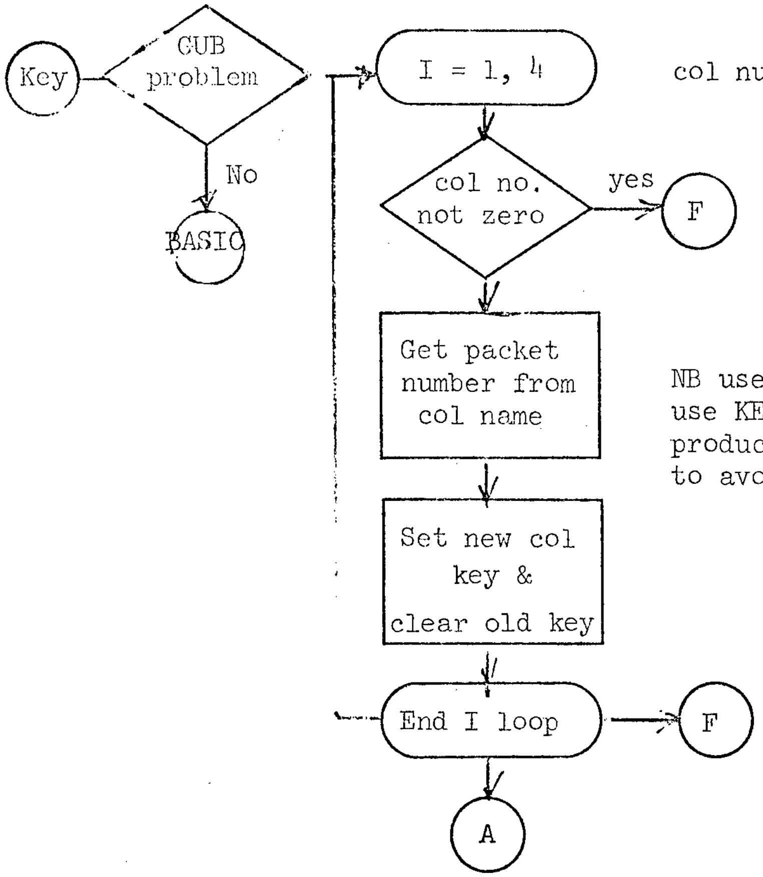
(1) MAPIN - Reads MAPIN cards from file IMAP and sets NAME record reads inverse from file INPUT to B.

(2) INMAP - Loads file IMAP from input card stream.


MAPIN
Print MAPIN and time using MESSG.
A
Read a card from file IMAP
user option
Card is "BASIC"
check card types and go to appropriate section
MAPOUT only
"KEY"
user option
"ATBND"
user option
"NULL"
B
page 2

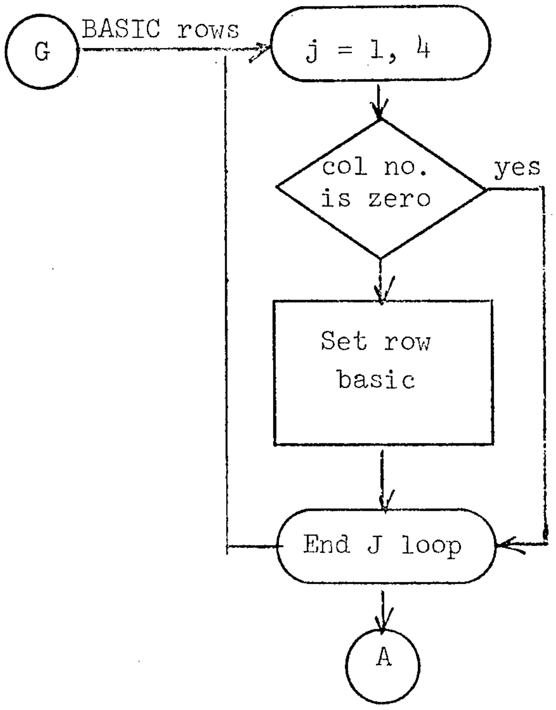

B
Mapout only
INVERS
added by INMAP
END
Return
Call error double print
Unrecognized type card
Return
ATBND
J = 1, 4
col. numbers on card
col no. is zero
D
Pick up column bound
bndj
finite
No
Note error
"ATBND col not bounded"
Set bound for column
dump bound data
CDC "PDUMP"
End of J loop
D
A


BASIC
"BASIC rows"
Yes
G
J = 1, 4
col no's/card
Col no. ≠ 0 set col basic
Update packet basic count in KEYS
if in a packet
End of J loop
A


Key
CUB problem
No
BASIC
I = 1, 4
col no. not zero
yes
F
Get packet number from col name
Set new col key & clear old key
End I loop
F
A

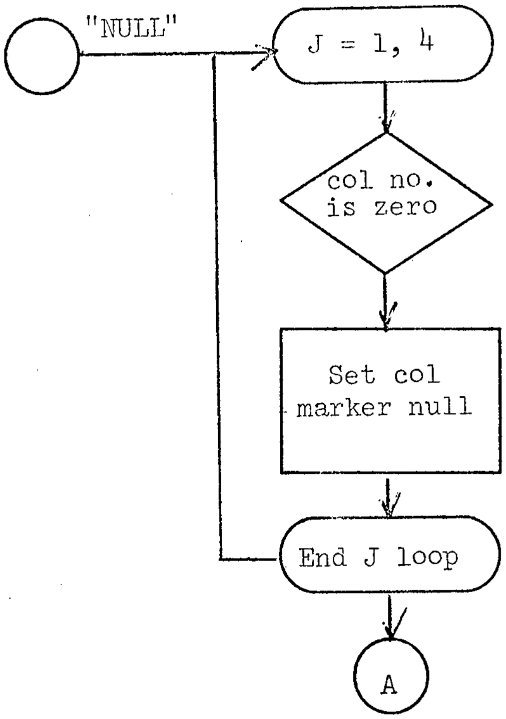
col numbers/mapin card

NB user should not
use KEY cards unless
produced by MAPOUT
to avoid column errors



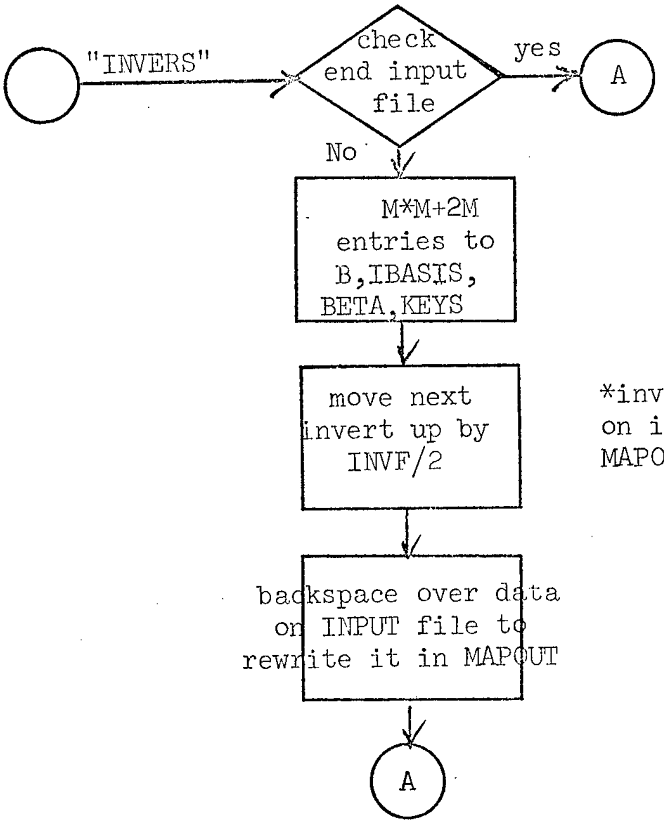
row no's/mapin card

NB row numbers
internally are not the
external no's
user should not use
this option manually,
MAPOUT - only

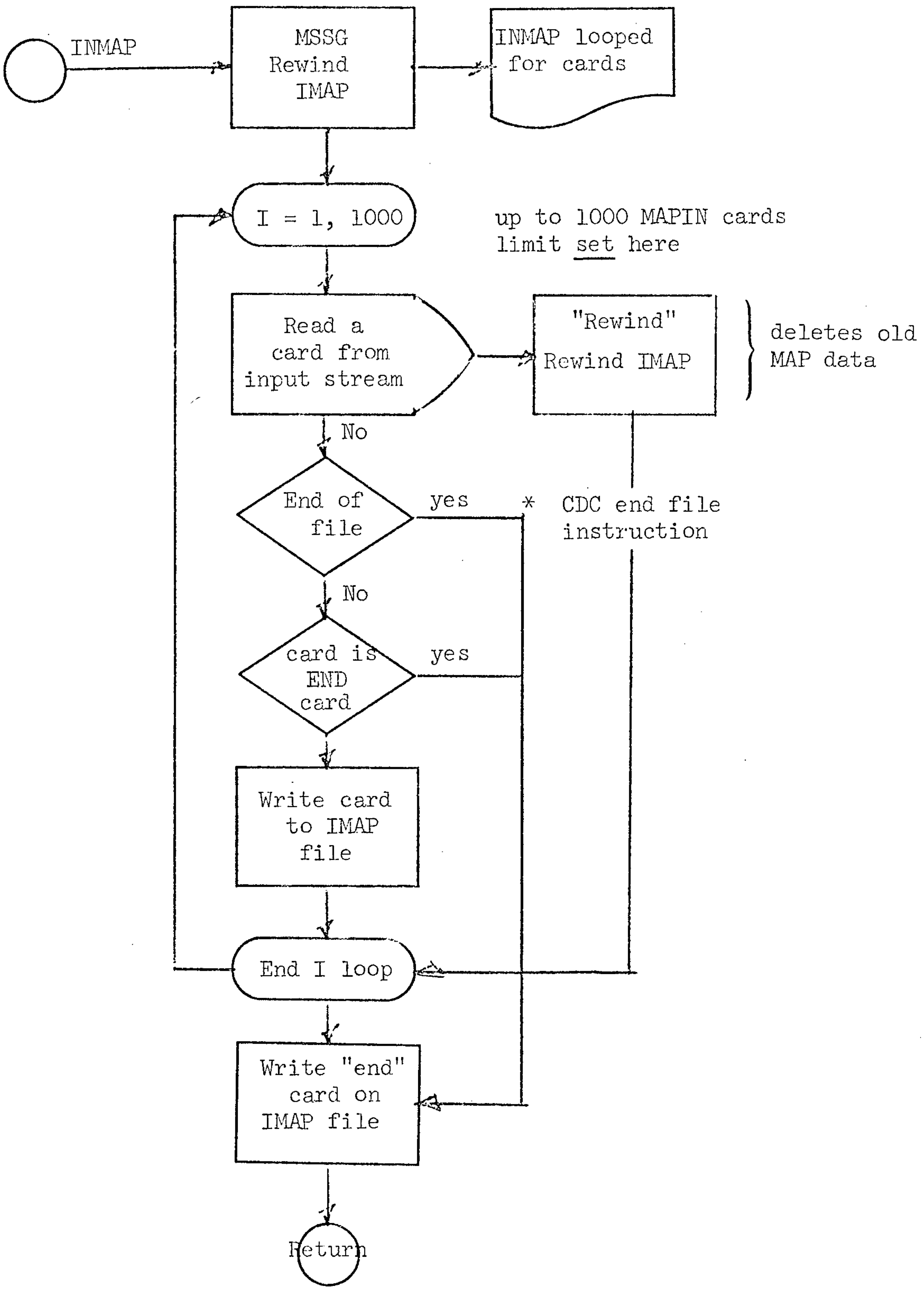


col no's/mapin card

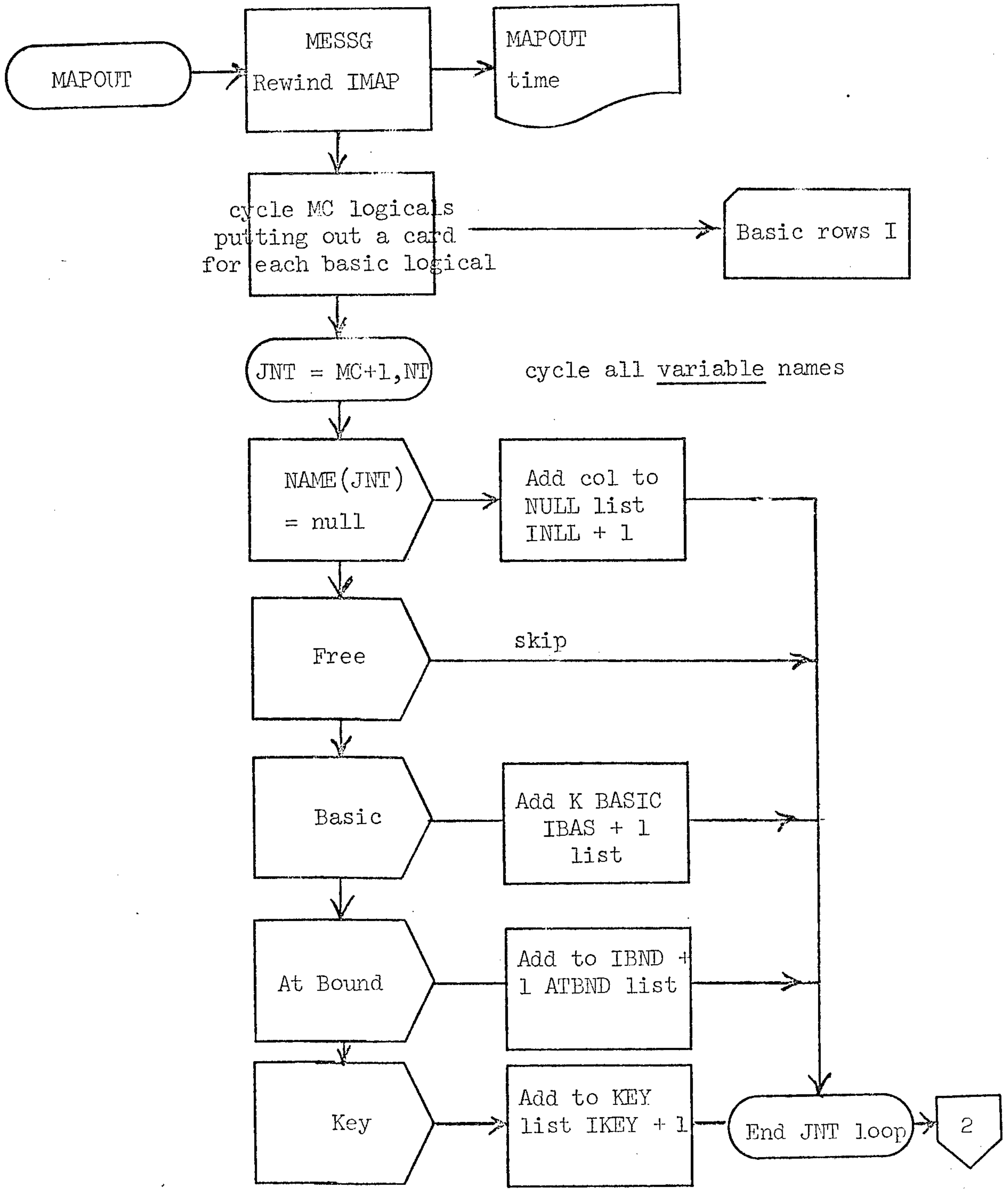
user option


"INVERS"
check end input file
yes
A
No
M*M+2M entries to B,IBASIS, BETA,KEYS
move next invert up by INVF/2
backspace over data on INPUT file to rewrite it in MAPOUT
A

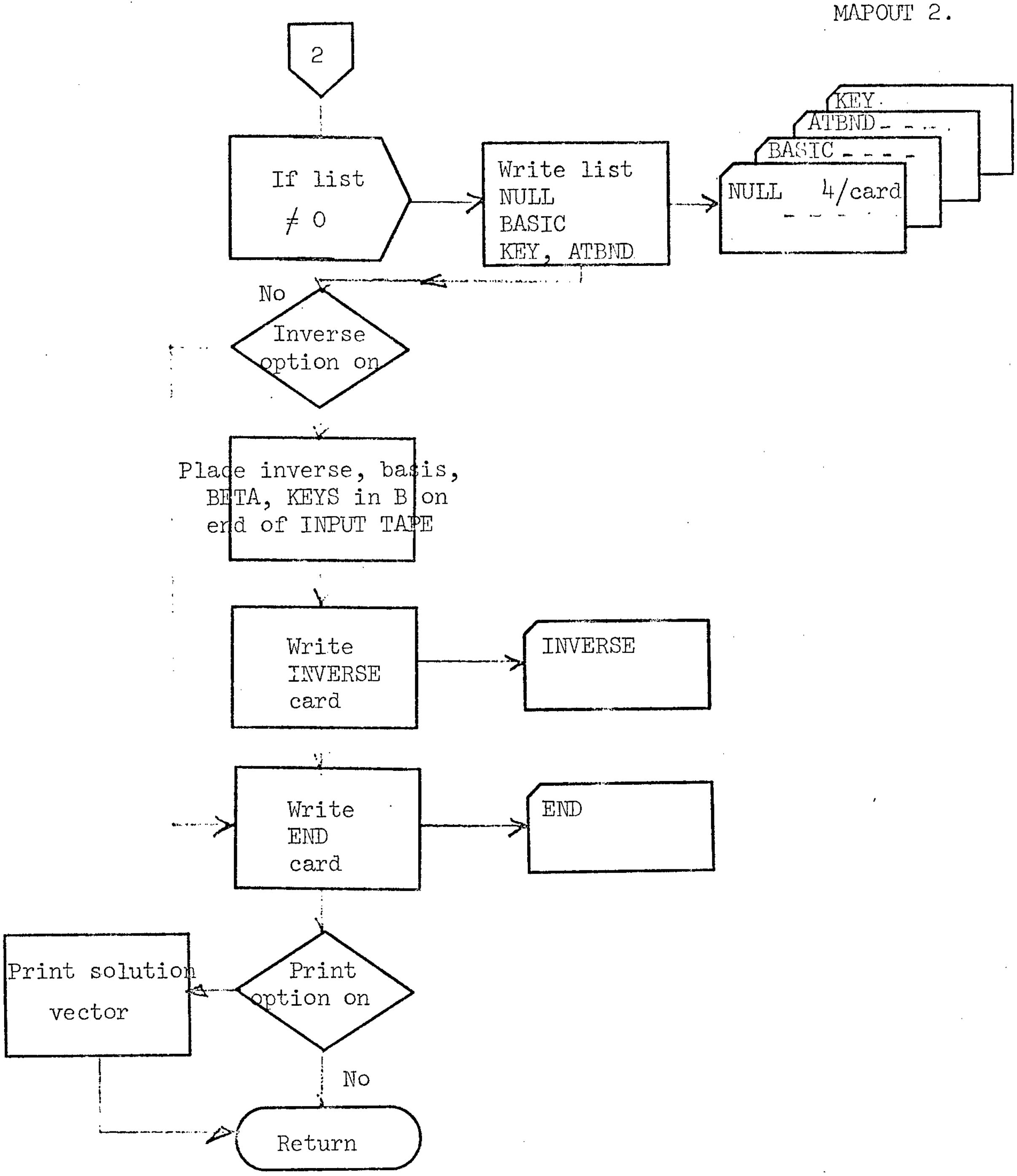
*inverse by rows placed on input file by MAPOUT


INMAP
MSSG
Rewind
IMAP
INMAP looped
for cards
I = 1, 1000
up to 1000 MAPIN cards
limit set here
Read a
card from
input stream
"Rewind"
Rewind IMAP
deletes old
MAP data
No
End of
file
yes
*
CDC end file
instruction
No
card is
END
card
yes
Write card
to IMAP
file
End I loop
Write "end"
card on
IMAP file
Return

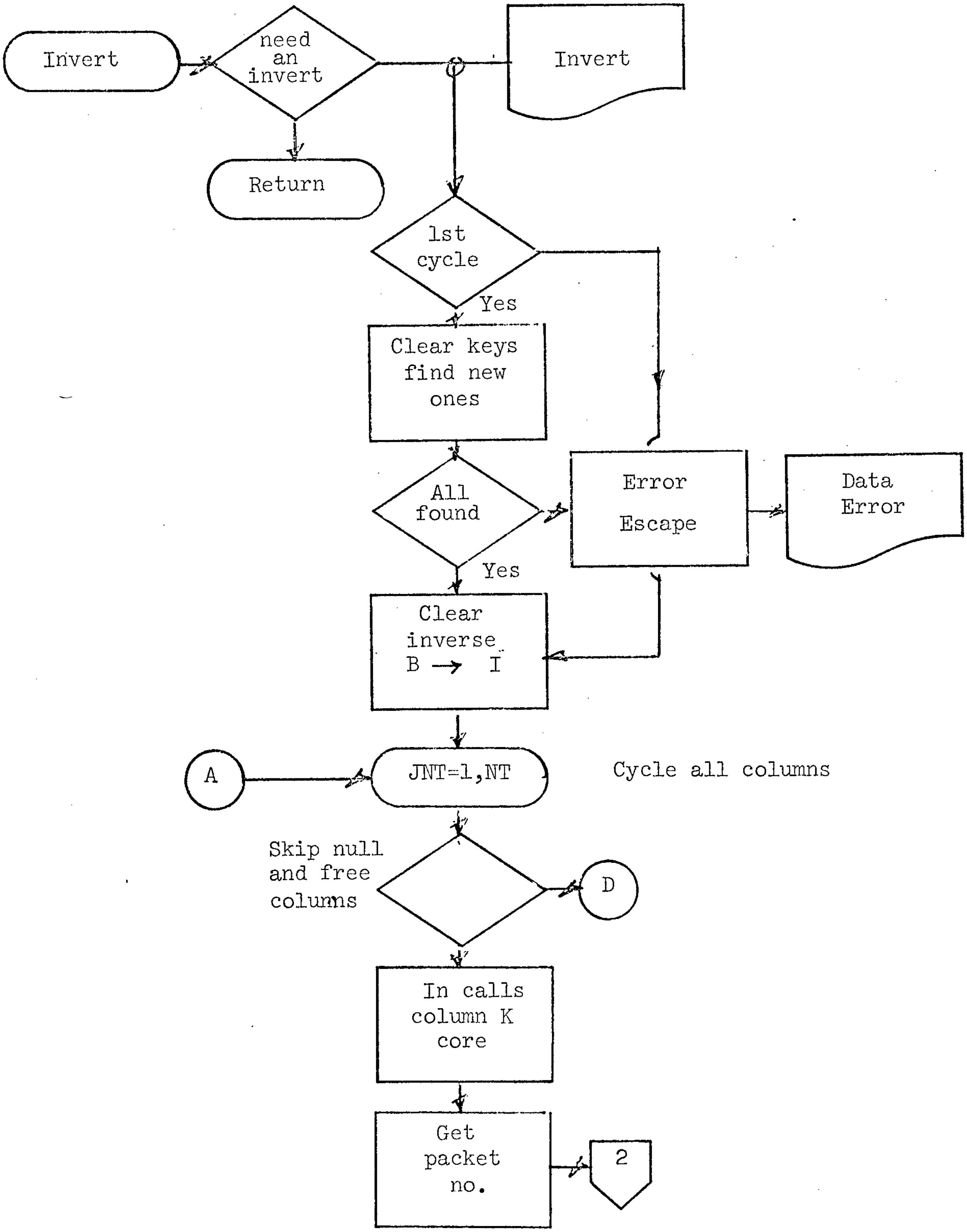
MAPOUT outputs on file IMAP a BCD card image definition of variables and inverse states compatible with MAPIN, whenever called and prints the current solution.

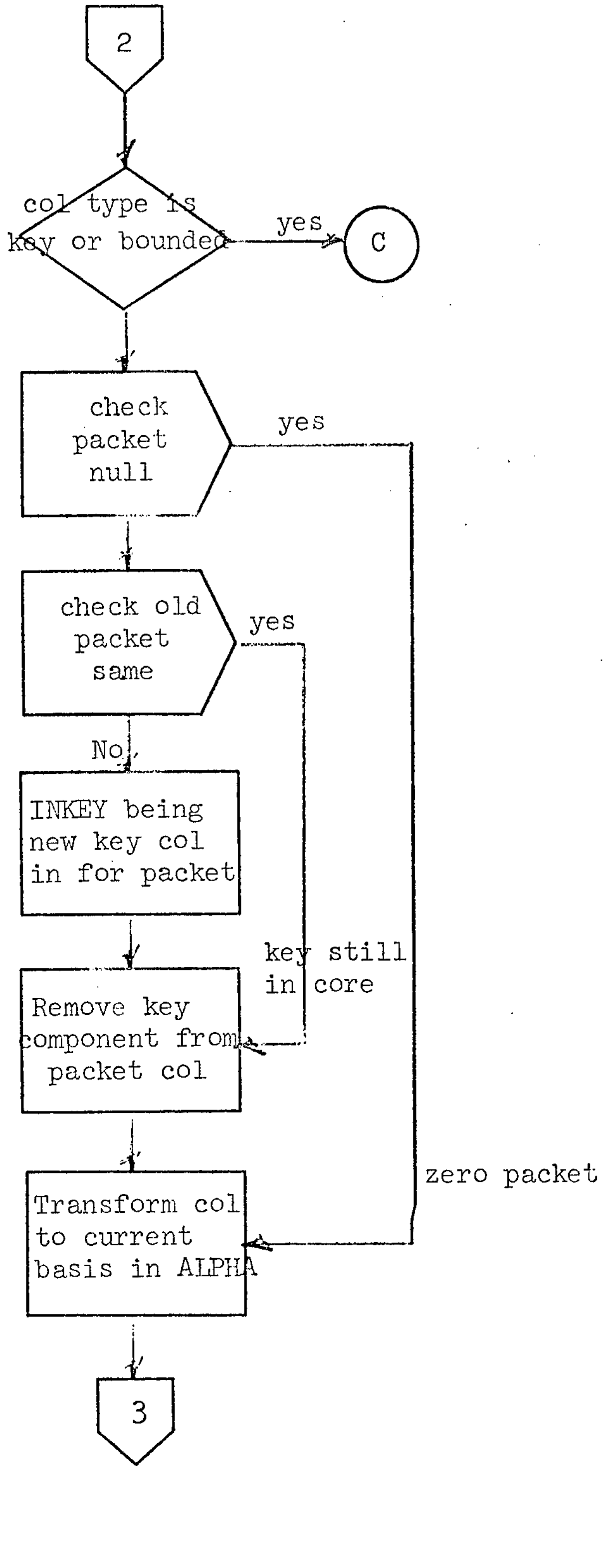


MAPOUT 2.

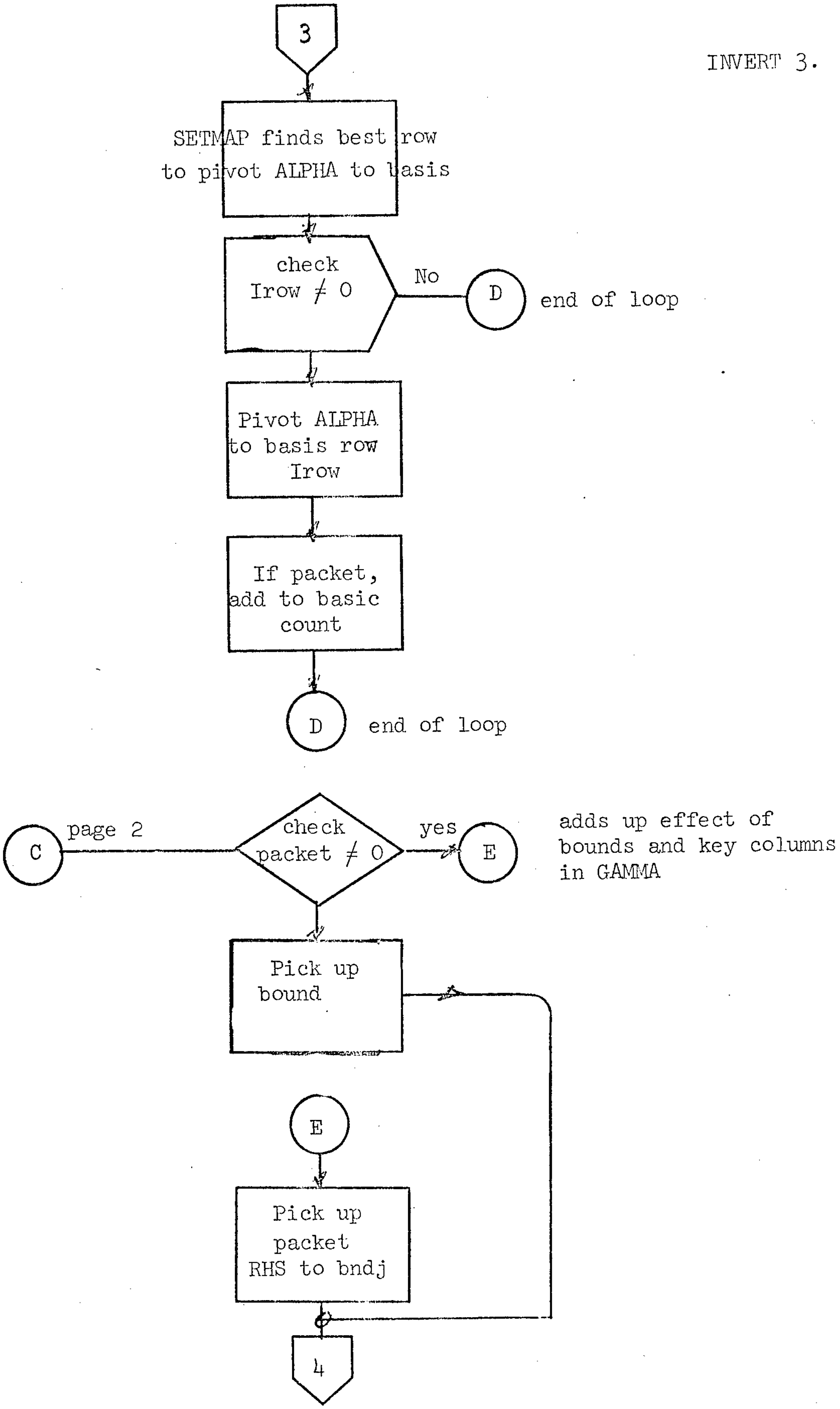


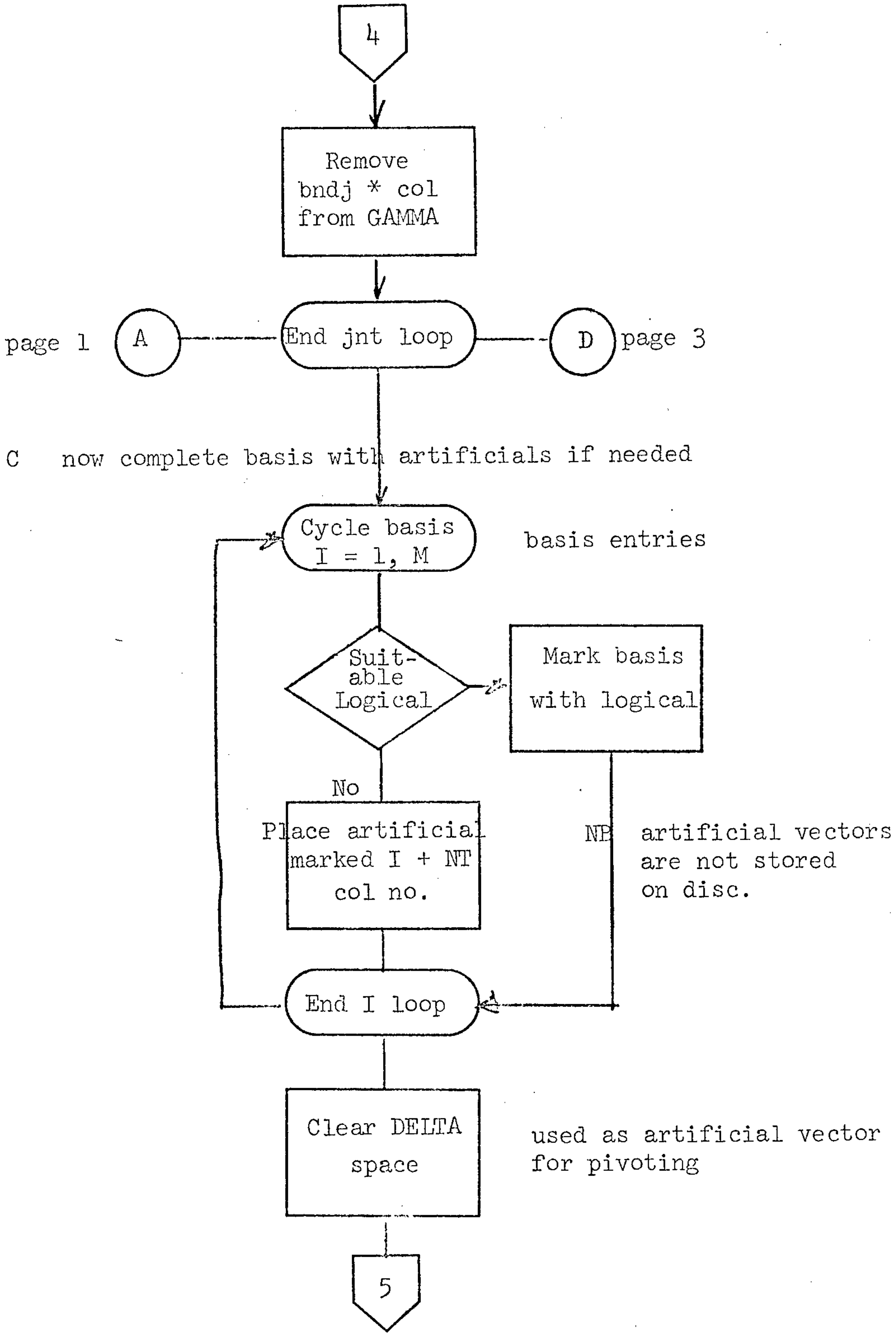
N.B. All MAPOUT cards can be generated manually. If they are inconsistant MAPIN uses the last setting of any column.


Invert
need an invert
Invert
Return
1st cycle
Yes
Clear keys find new ones
All found
Yes
Error Escape
Data Error
Clear inverse B → I
A
JNT=1,NT
Cycle all columns
Skip null and free columns
D
In calls column K core
Get packet no.
2


2
col type is key or bounded
yes
C
check packet null
yes
check old packet same
yes
No
INKEY being new key col in for packet
key still in core
Remove key component from packet col
zero packet
Transform col to current basis in ALPHA
3

INVERT 3.


3
SETMAP finds best row to pivot ALPHA to basis
check Irow ≠ 0
No
D
end of loop
Pivot ALPHA to basis row Irow
If packet, add to basic count
D
end of loop
C
page 2
check packet ≠ 0
yes
E
adds up effect of bounds and key columns in GAMMA
Pick up bound
E
Pick up packet RHS to bndj
4

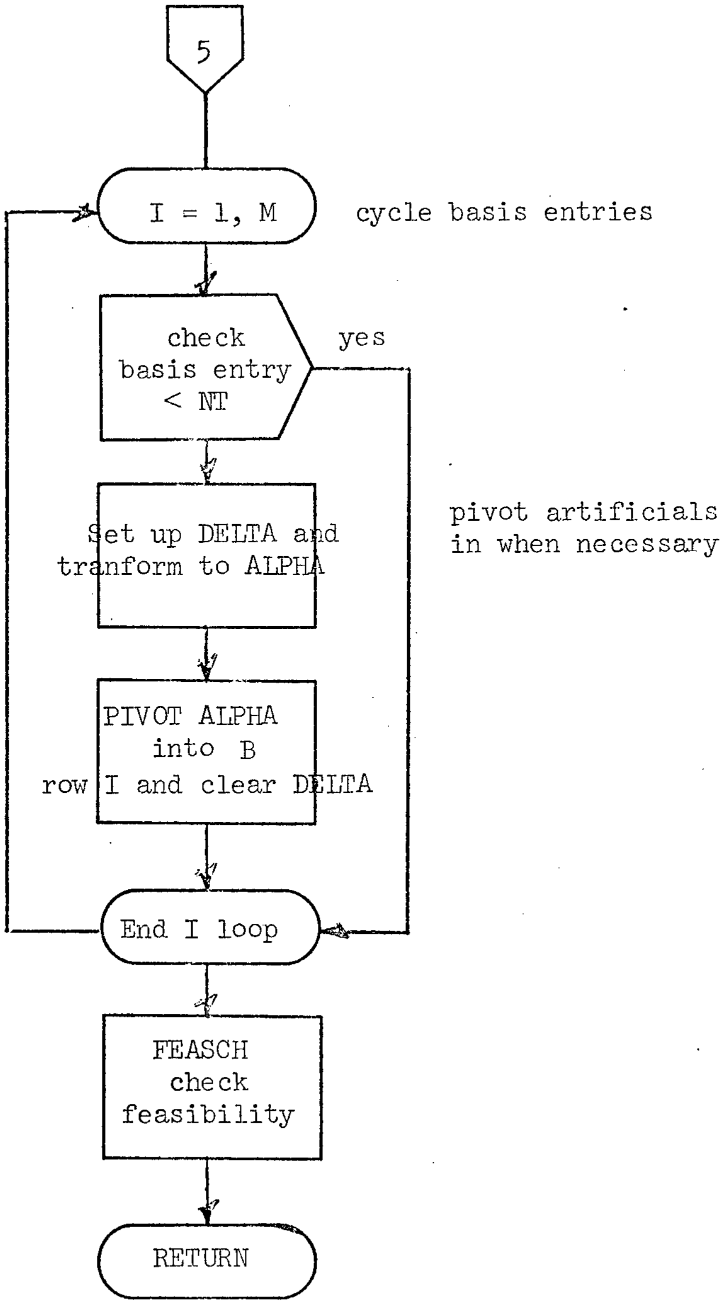


C now complete basis with artificials if needed

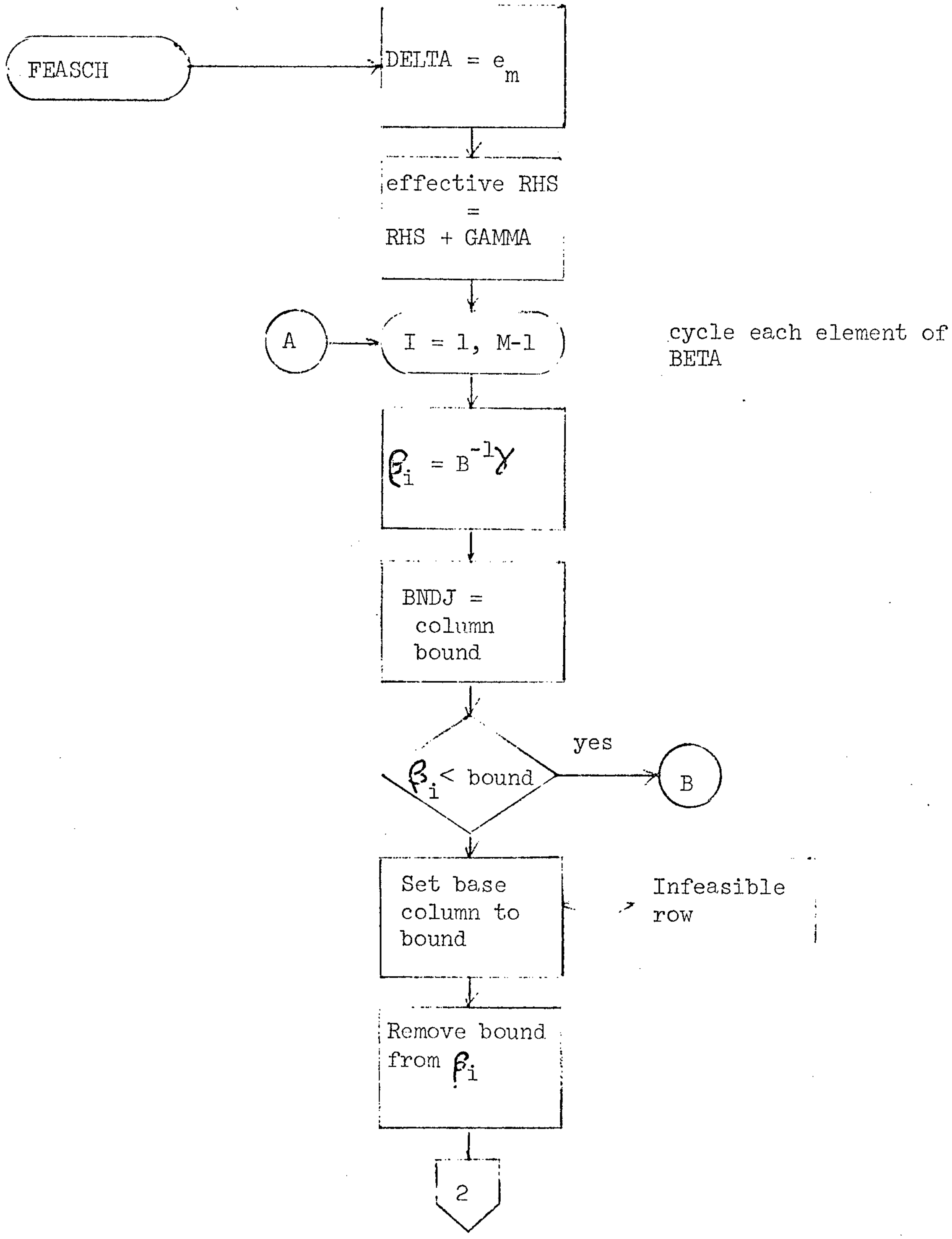
basis entries

artificial vectors are not stored on disc.

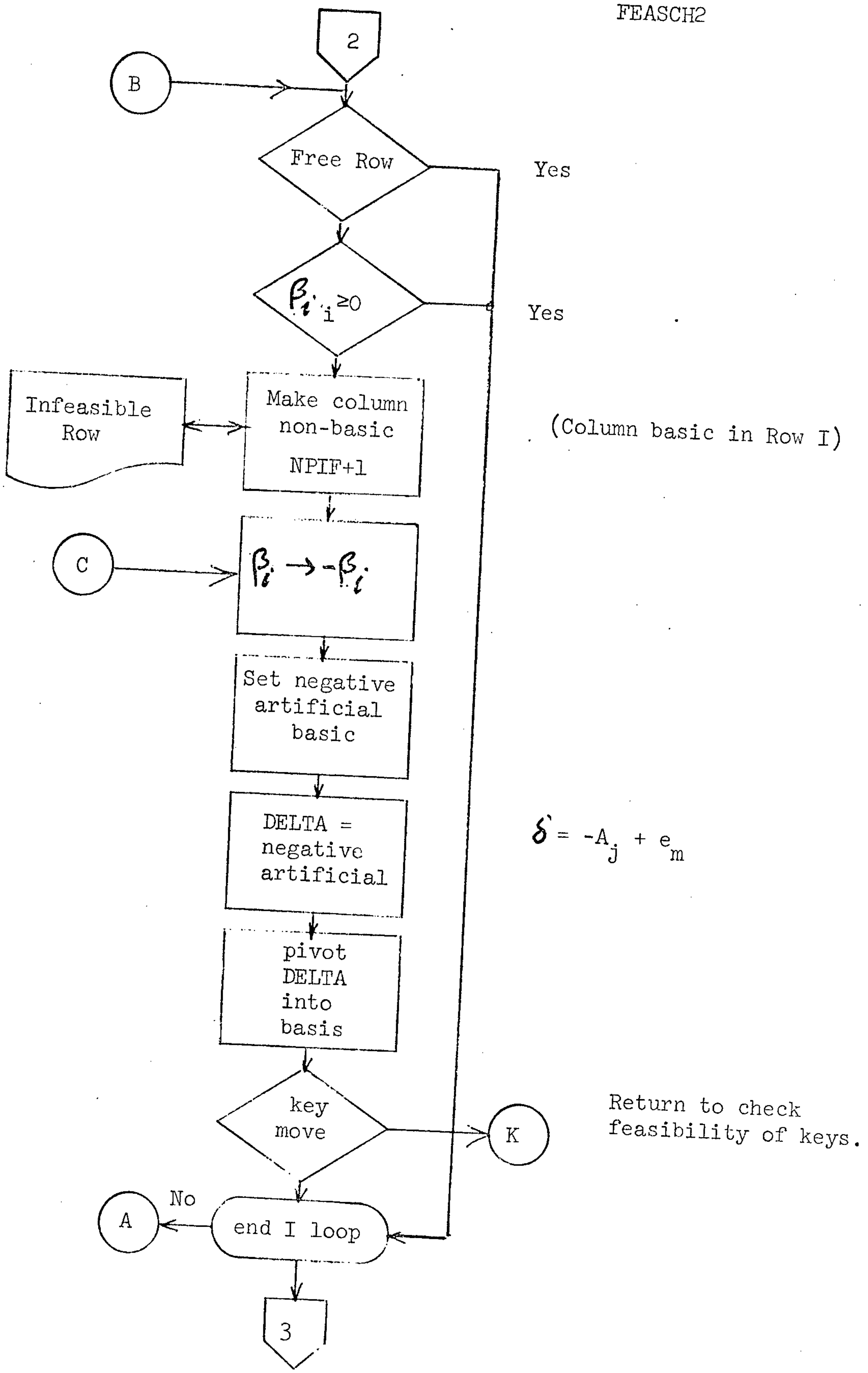
used as artificial vector for pivoting

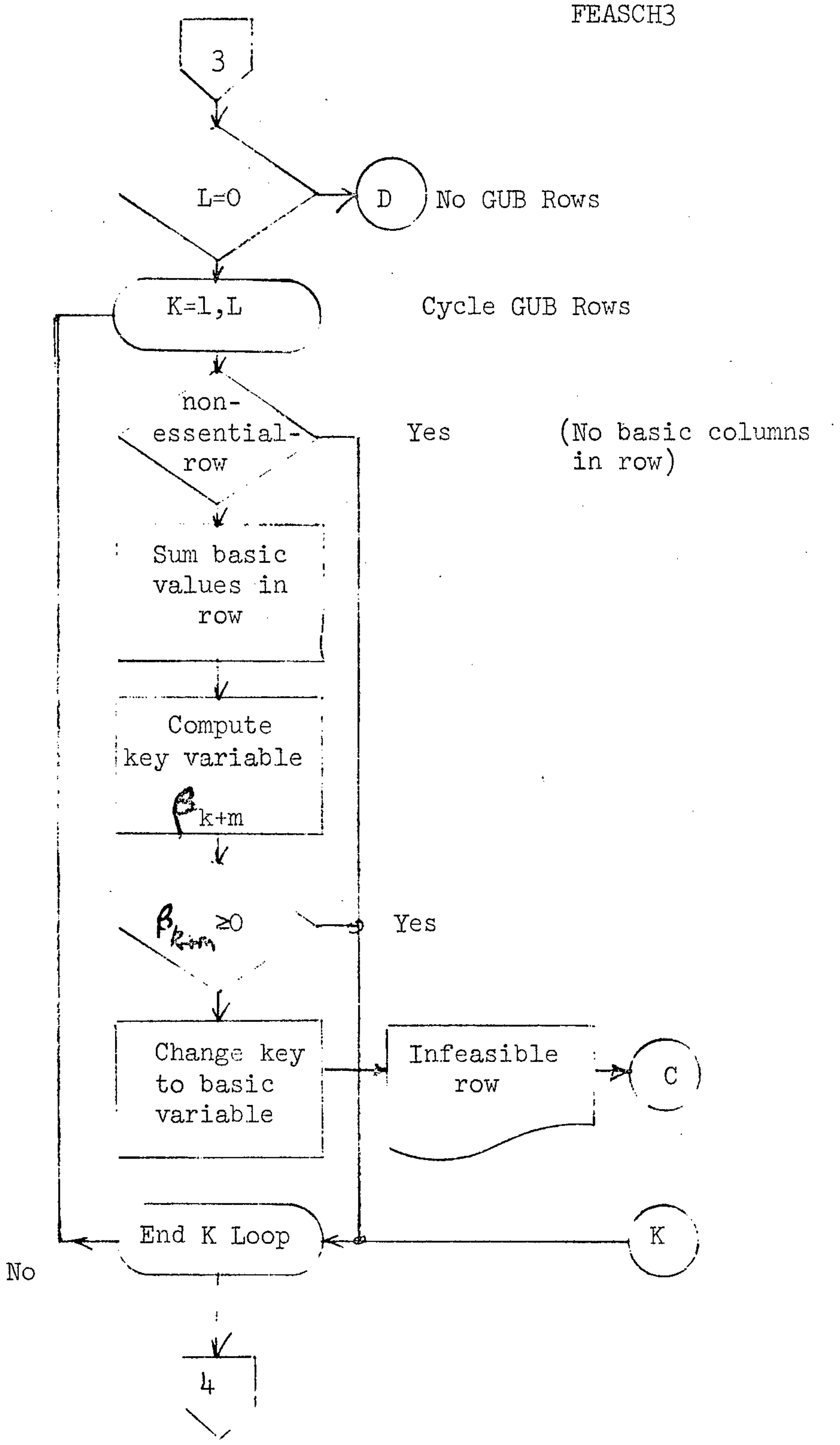

5
I = 1, M
cycle basis entries
check basis entry < NT
yes
Set up DELTA and tranform to ALPHA
pivot artificials in when necessary
PIVOT ALPHA into B row I and clear DELTA
End I loop
FEASCH check feasibility
RETURN

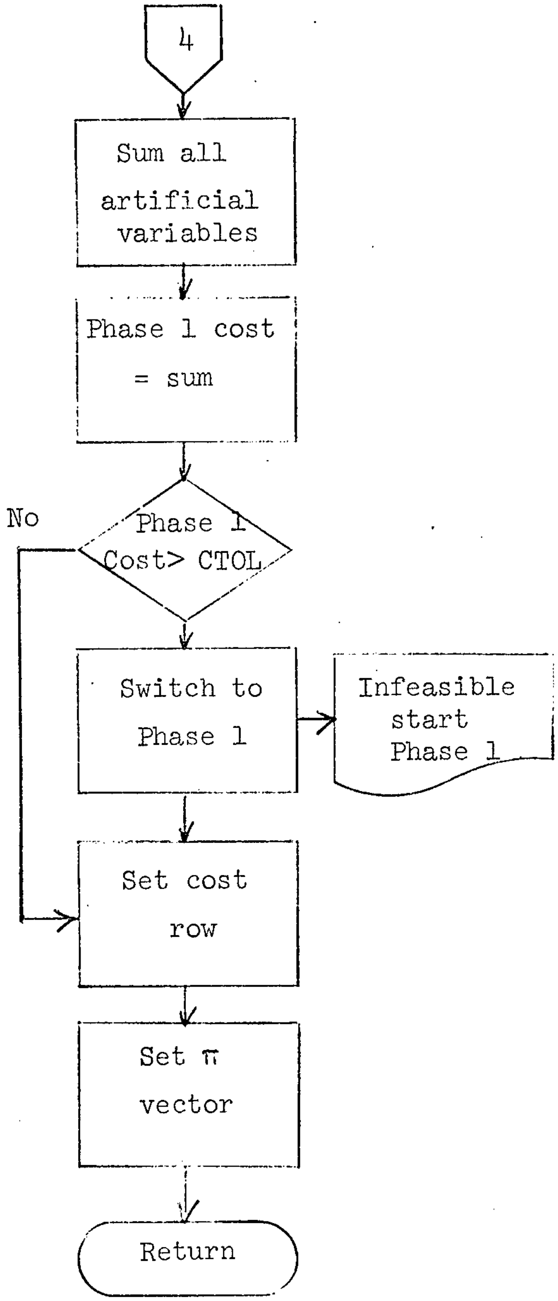
This routine computes the solution and checks feasibility.



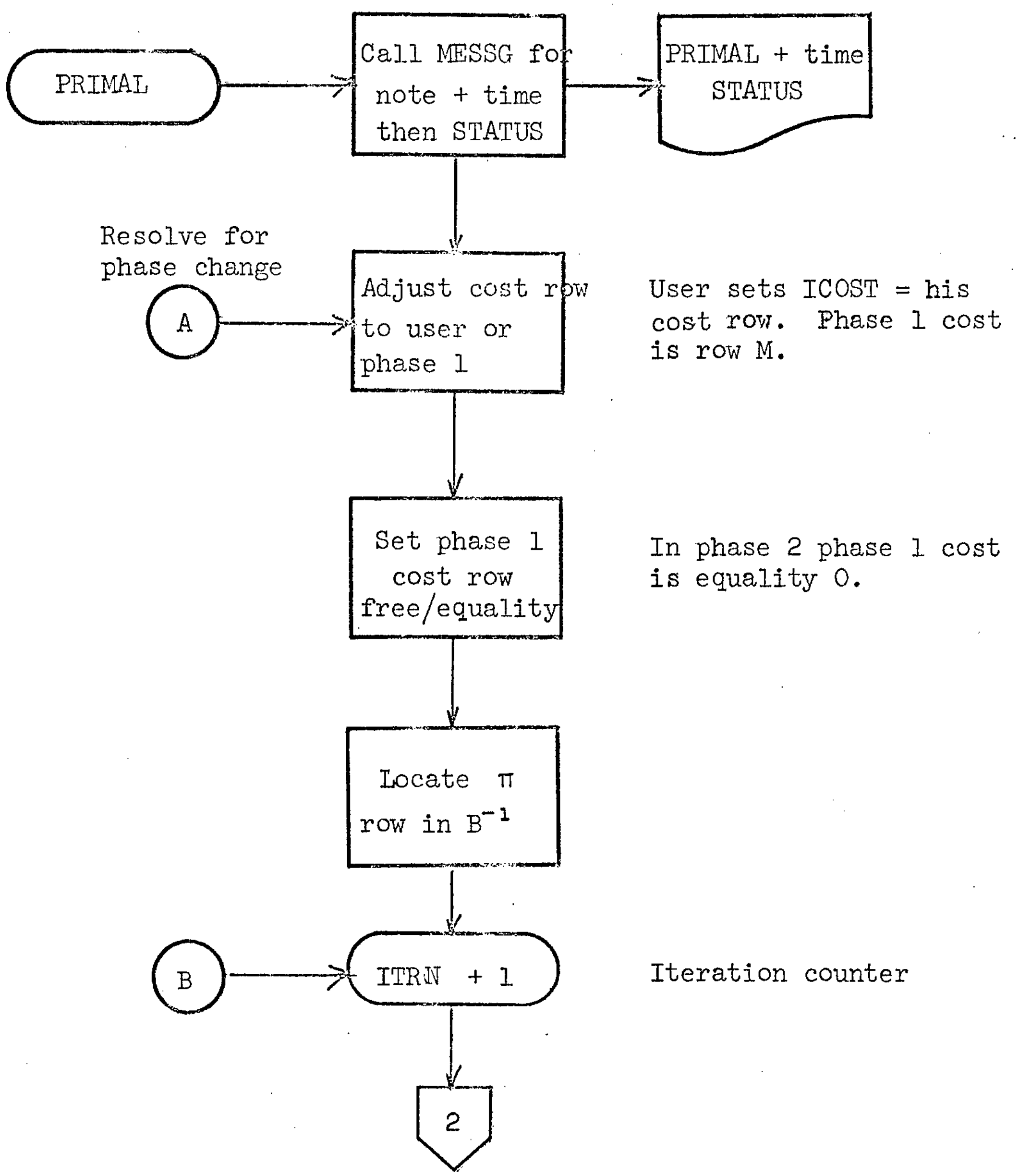
FEASCH2

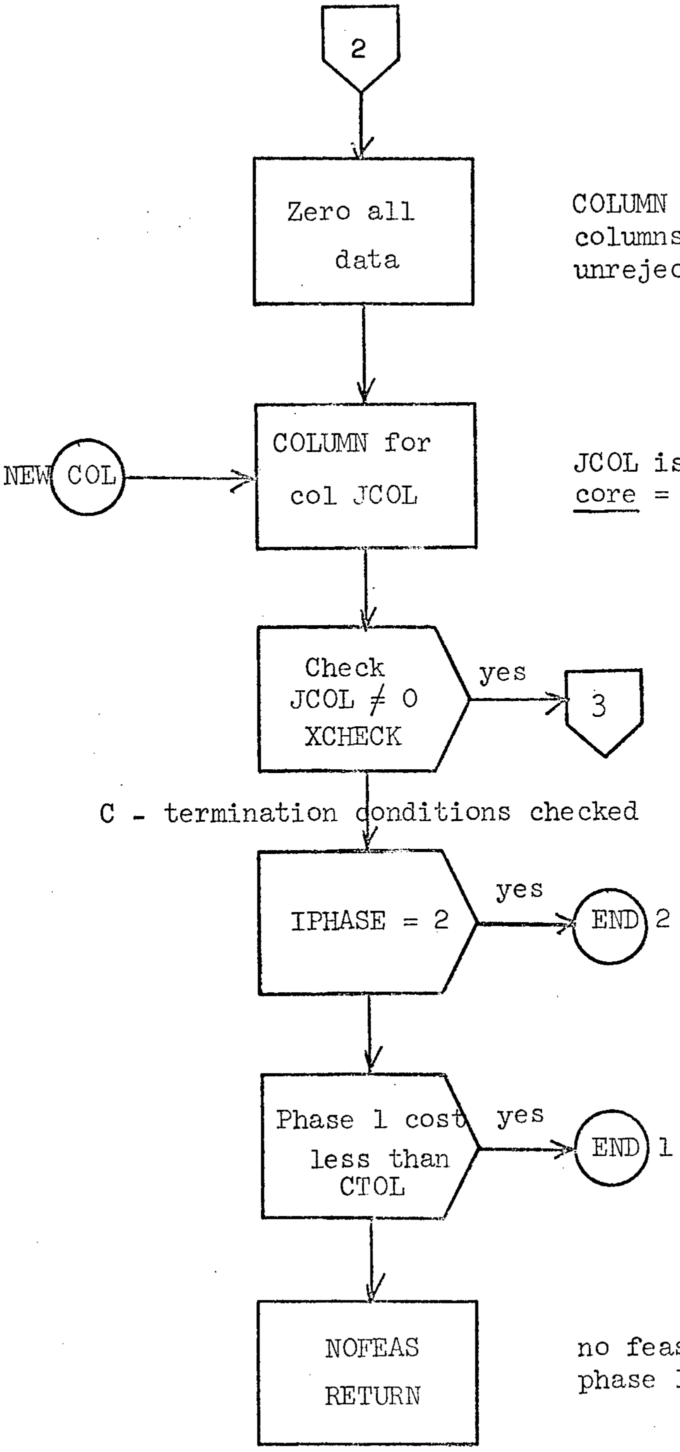

2
B
Free Row
Yes
β_i $_i \geq 0$
Yes
Infeasible Row
Make column non-basic
NPIF+1
(Column basic in Row I)
C
$\beta_i \rightarrow -\beta_i$
Set negative artificial basic
DELTA = negative artificial
$\delta = -A_j + e_m$
pivot DELTA into basis
key move
K
Return to check feasibility of keys.
No
A
end I loop
3


3
L=0
D
No GUB Rows
K=1,L
Cycle GUB Rows
non-essential-row
Yes
(No basic columns in row)
Sum basic values in row
Compute key variable β_{k+m}
$\beta_{k+m} \geq 0$
Yes
Change key to basic variable
Infeasible row
C
End K Loop
K
No
4


4
Sum all artificial variables
Phase 1 cost = sum
Phase 1 Cost> CTOL
No
Switch to Phase 1
Infeasible start Phase 1
Set cost row
Set π vector
Return

PRIMAL runs the 2 phase revised simplex algorithm from both phases and exits via EXIT.


PRIMAL
Call MESSG for note + time then STATUS
PRIMAL + time STATUS
Resolve for phase change
A
Adjust cost row to user or phase 1
User sets ICOST = his cost row. Phase 1 cost is row M.
Set phase 1 cost row free/equality
In phase 2 phase 1 cost is equality 0.
Locate π row in B^{-1}
B
ITRN + 1
Iteration counter
2



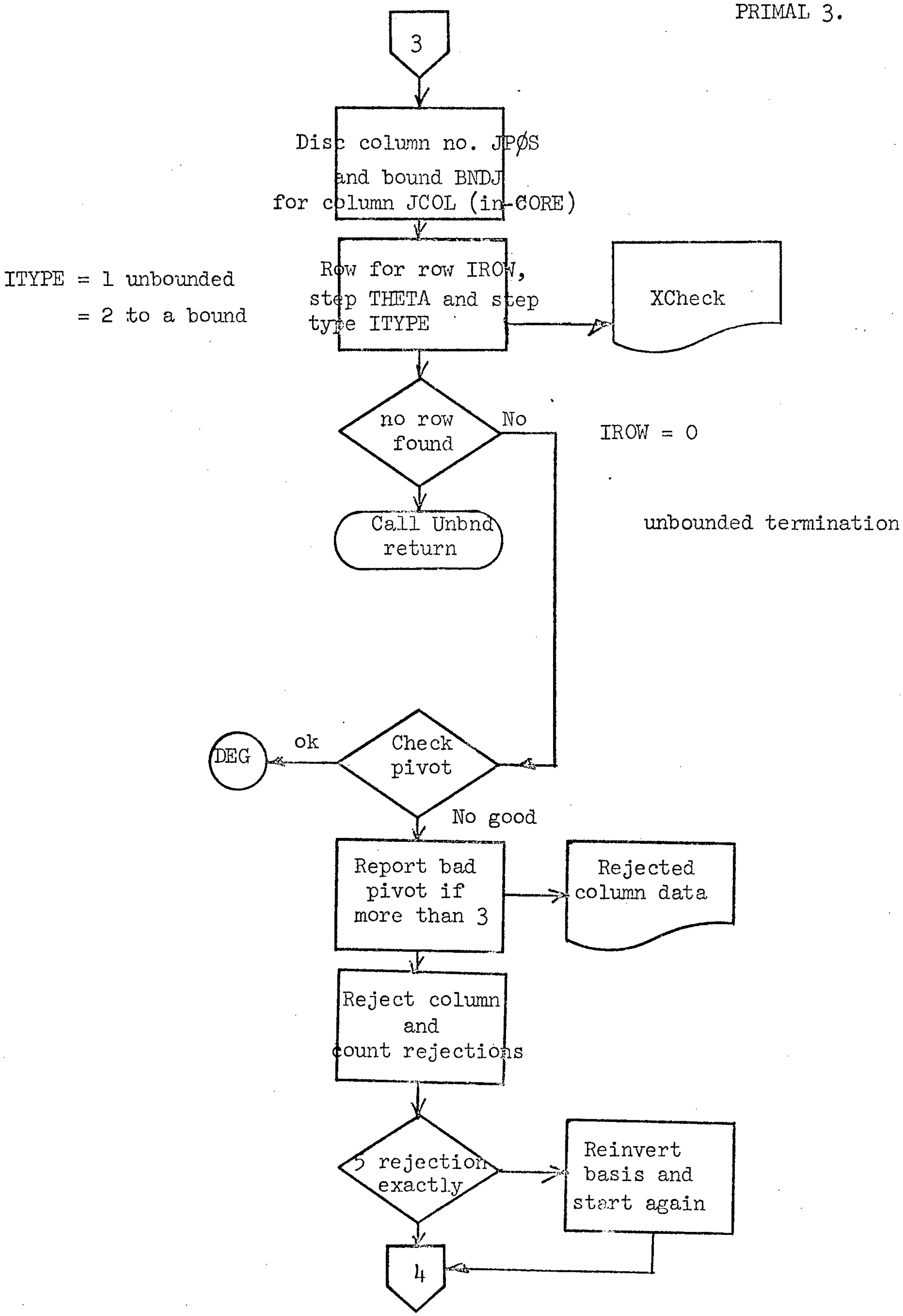
COLUMN prices out in-core columns and finds best unrejected column .

JCOL is location in <u>core</u> = 1 to JNCORE

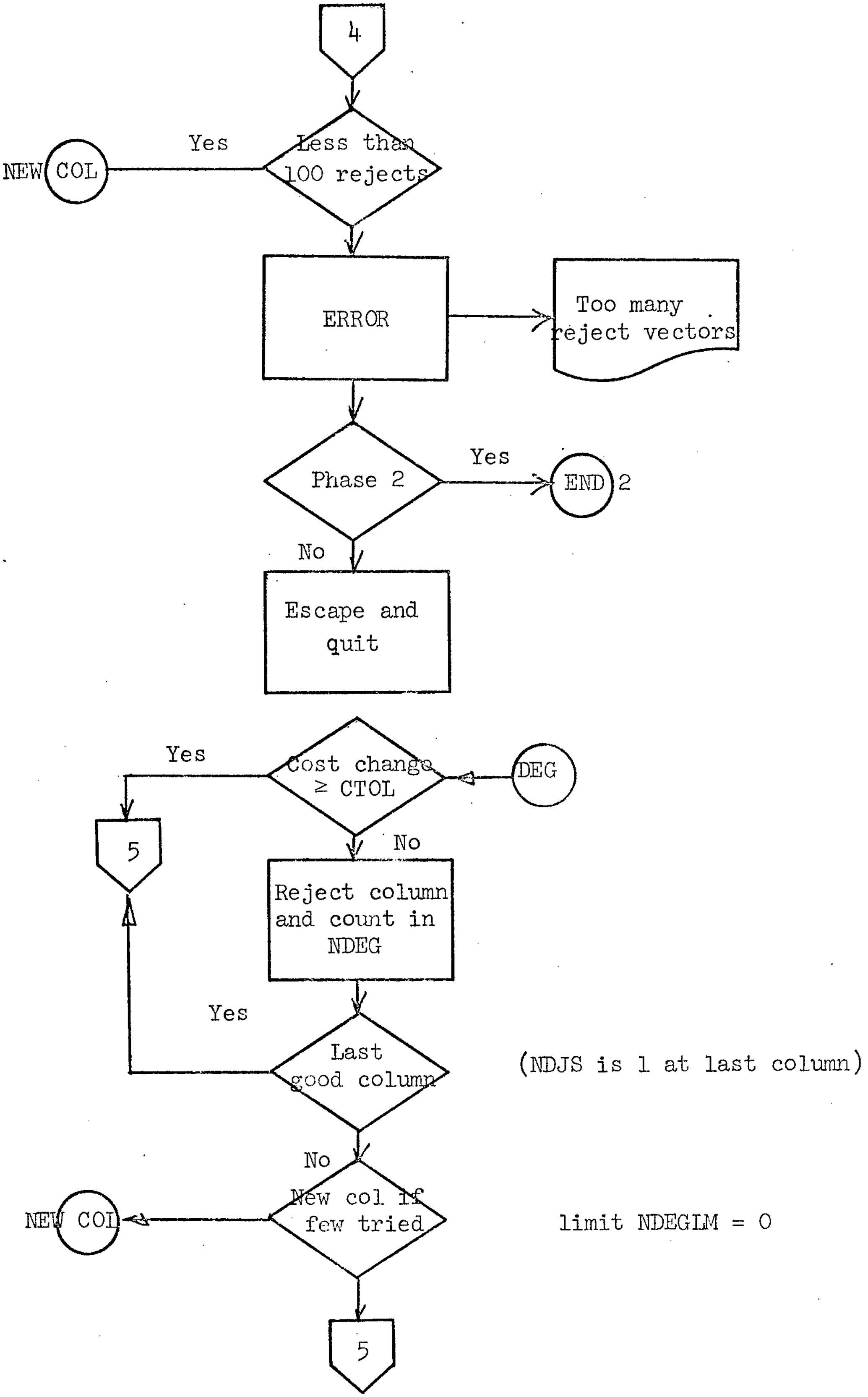
Optimum phase 2 ending

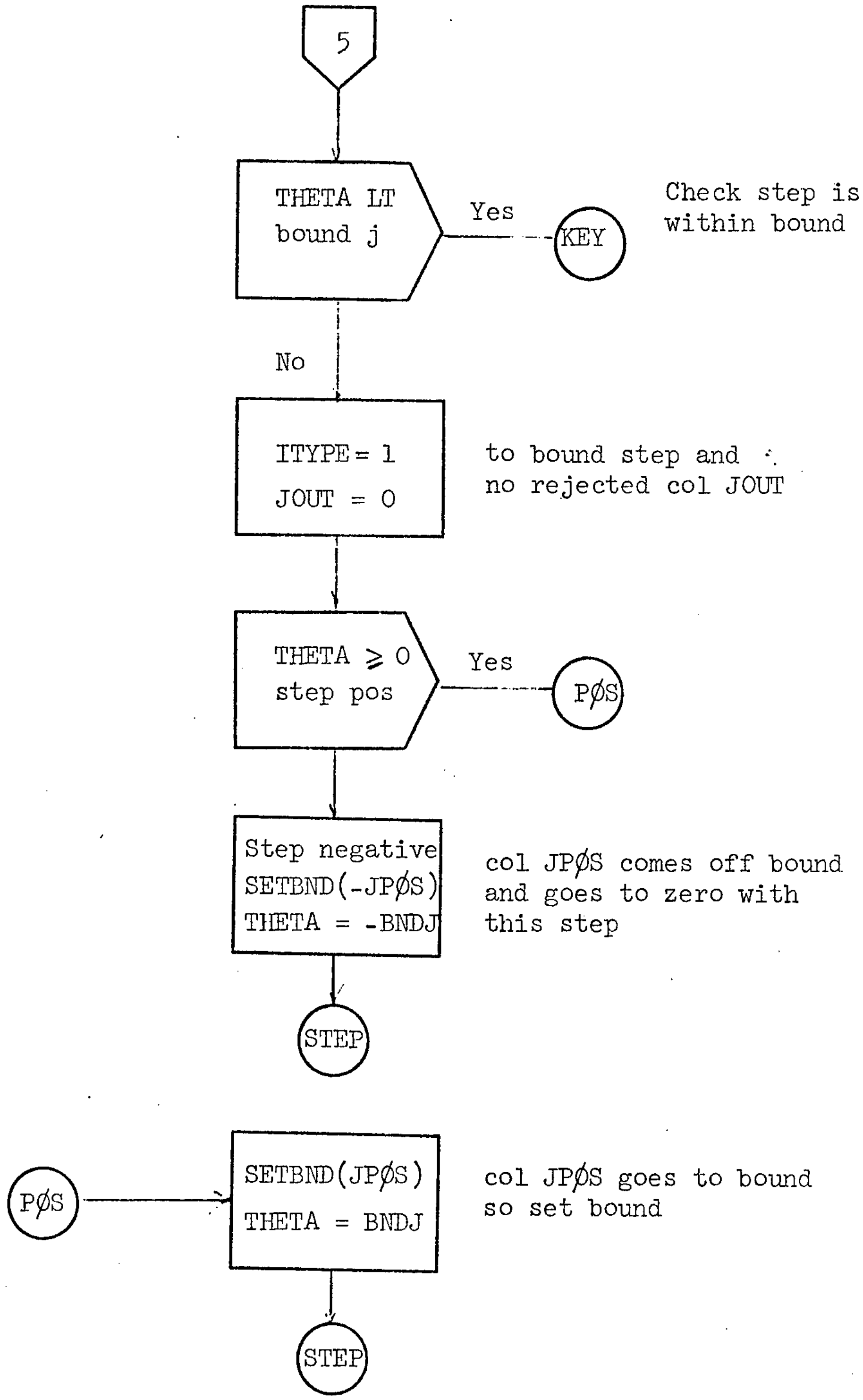
Optimum phase 1 ending.

no feasible solution since phase 1 cost not zero


3
Disc column no. JPØS
and bound BNDJ
for column JCOL (in-CORE)
ITYPE = 1 unbounded
= 2 to a bound
Row for row IROW,
step THETA and step
type ITYPE
XCheck
no row
found
No
IROW = 0
Call Unbnd
return
unbounded termination
DEG
ok
Check
pivot
No good
Report bad
pivot if
more than 3
Rejected
column data
Reject column
and
count rejections
5 rejection
exactly
Reinvert
basis and
start again
4

PRIMAL 4.


4
Less than 100 rejects
Yes
NEW COL
ERROR
Too many reject vectors
Phase 2
Yes
END 2
No
Escape and quit
Cost change ≥ CTOL
DEG
Yes
5
No
Reject column and count in NDEG
Yes
Last good column
(NDJS is 1 at last column)
No
New col if few tried
NEW COL
limit NDEGLM = 0
5

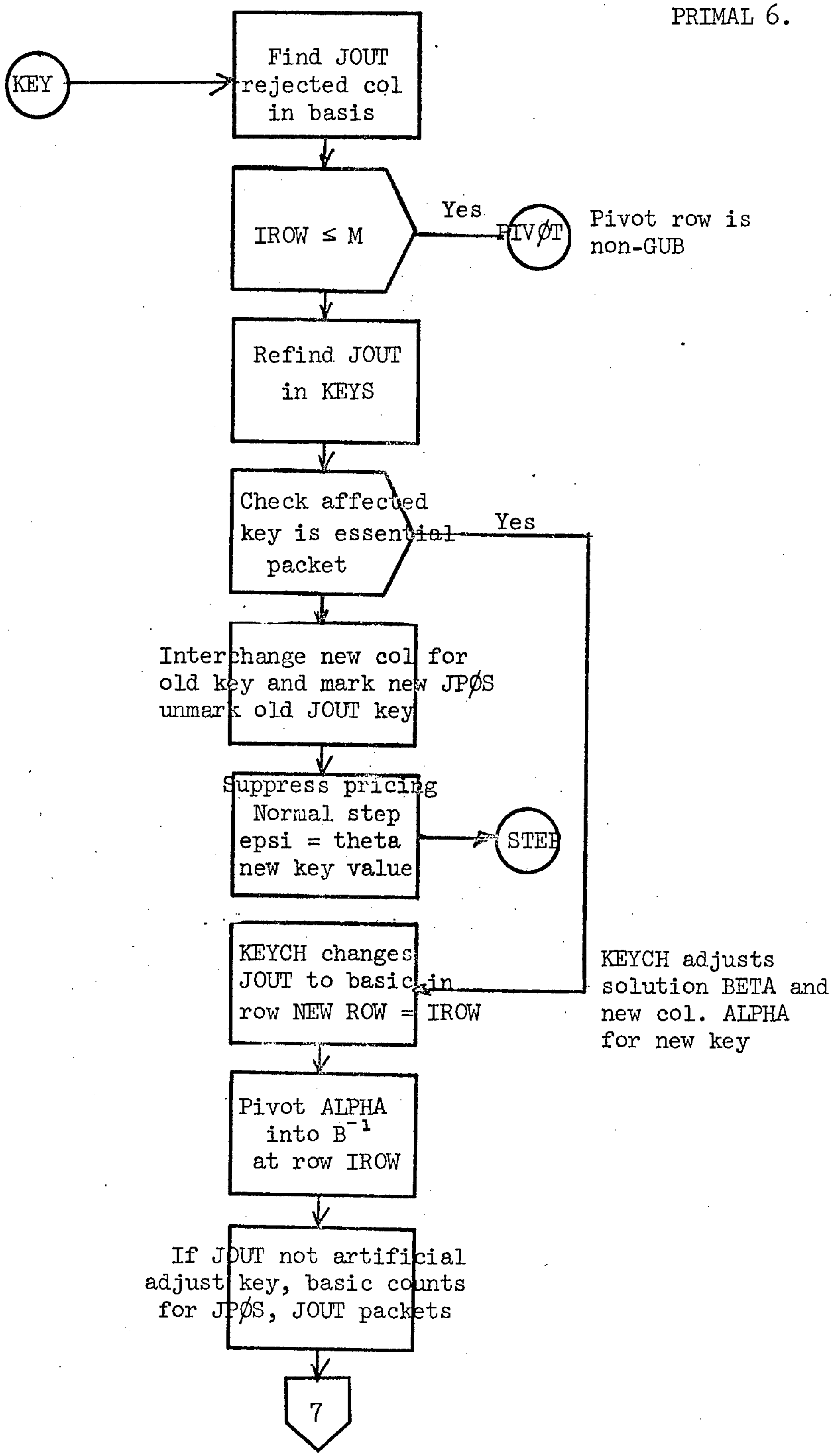


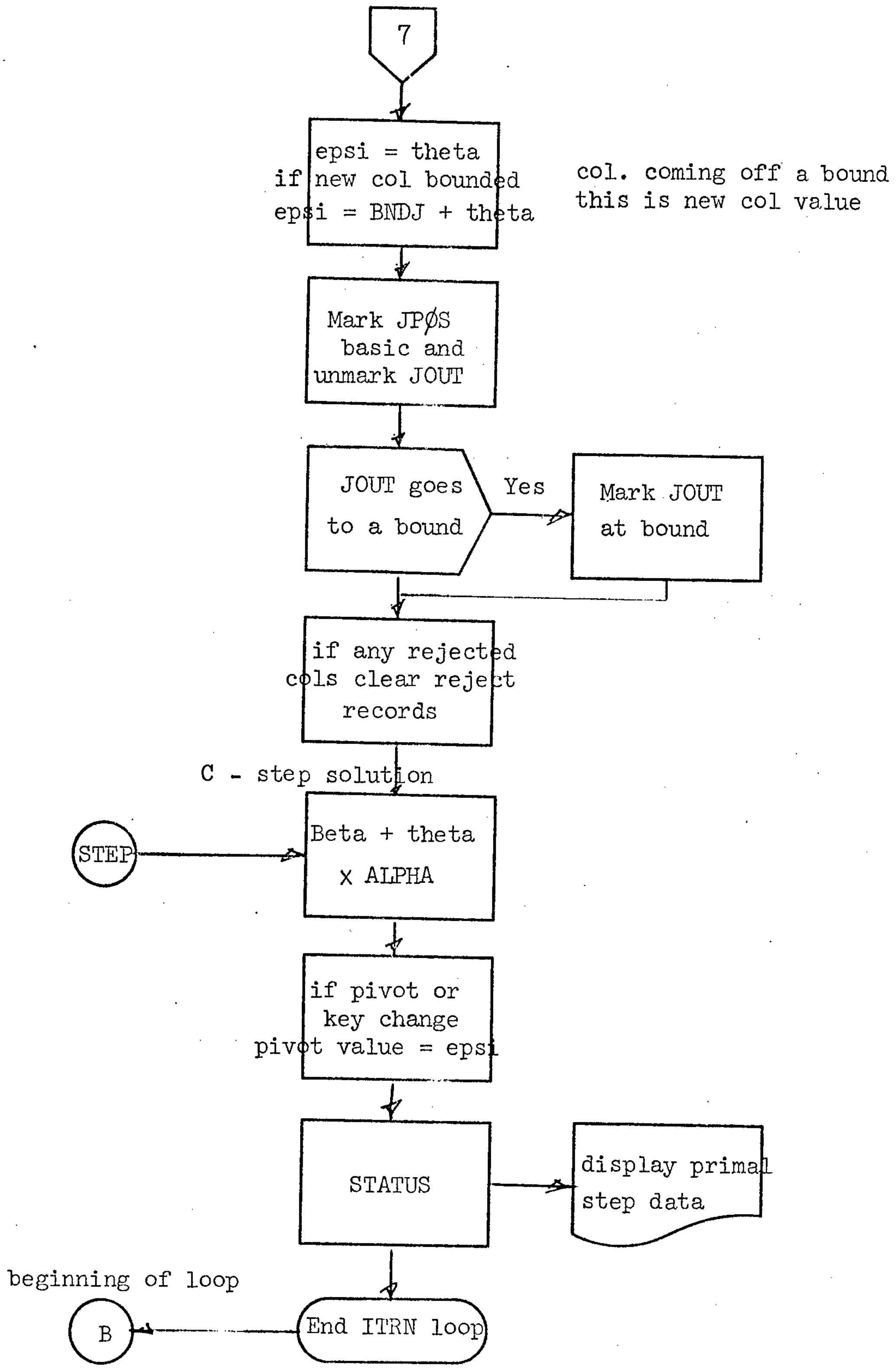
Check step is within bound

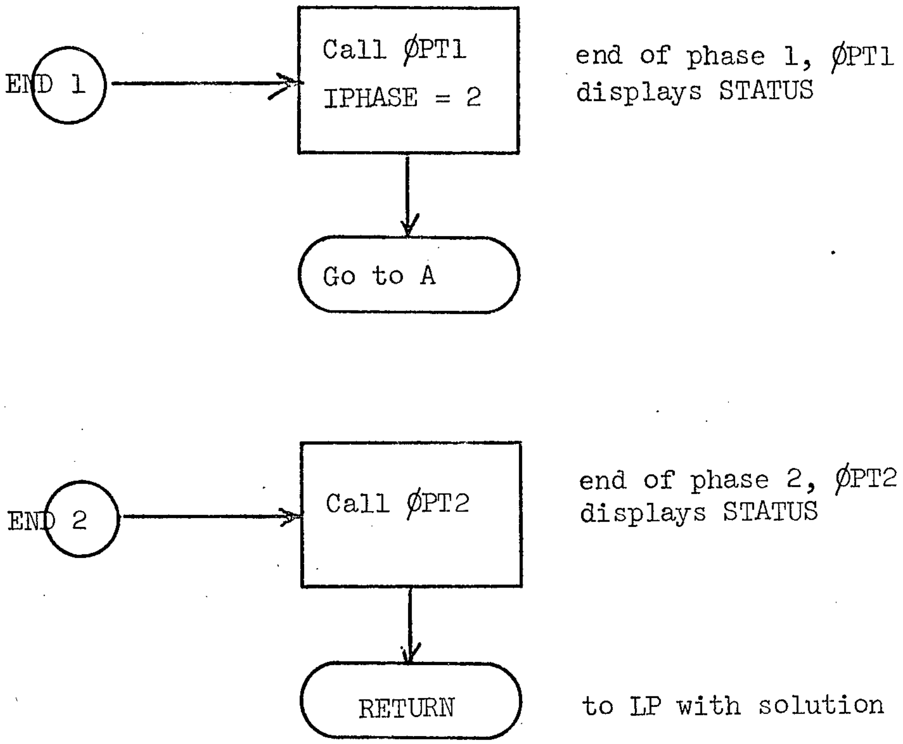
to bound step and no rejected col JOUT

col JPØS comes off bound and goes to zero with this step

col JPØS goes to bound so set bound

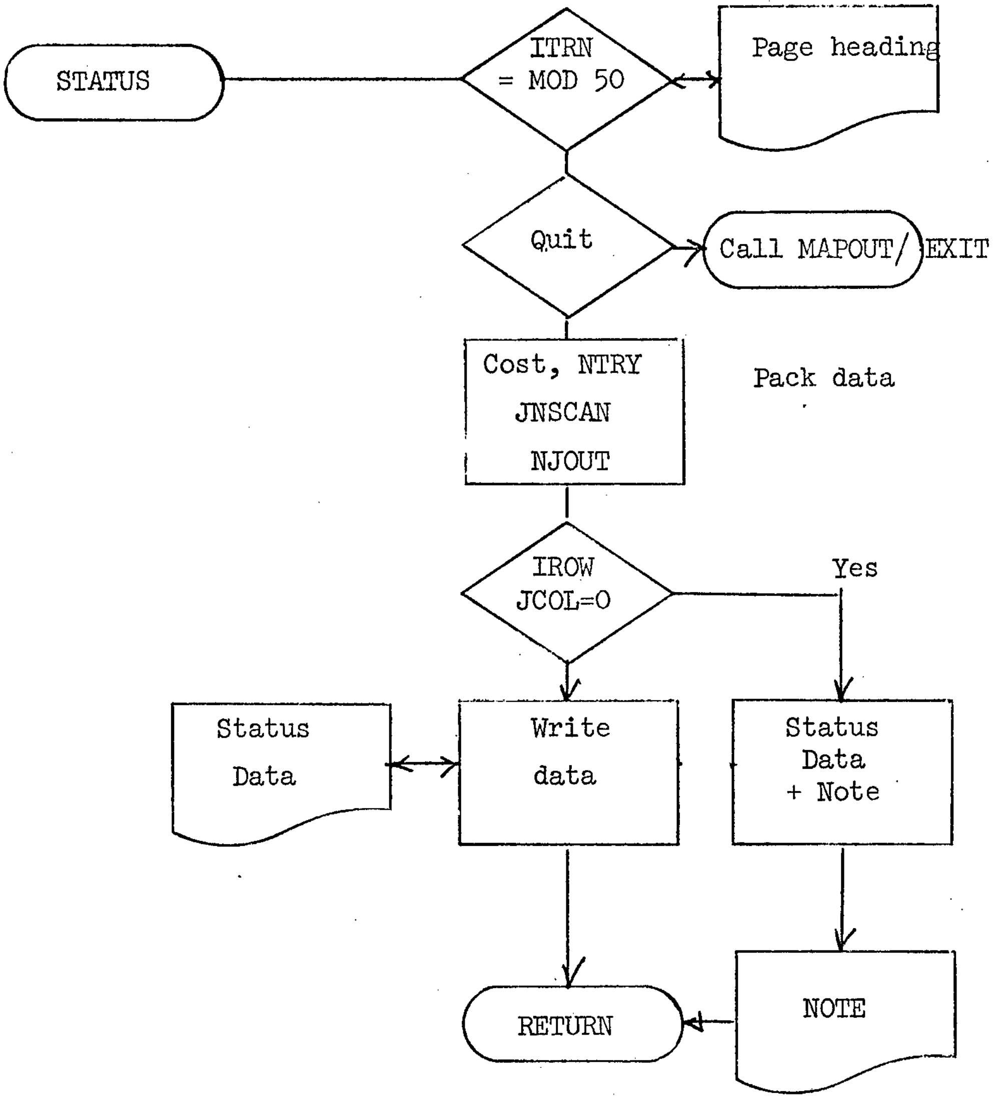

KEY
Find JOUT rejected col in basis
IROW ≤ M
Yes
PIVØT
Pivot row is non-GUB
Refind JOUT in KEYS
Check affected key is essential packet
Yes
Interchange new col for old key and mark new JPØS unmark old JOUT key
Suppress pricing Normal step epsi = theta new key value
STEP
KEYCH changes JOUT to basic in row NEW ROW = IROW
KEYCH adjusts solution BETA and new col. ALPHA for new key
Pivot ALPHA into B⁻¹ at row IROW
If JOUT not artificial adjust key, basic counts for JPØS, JOUT packets
7


7
epsi = theta
if new col bounded
epsi = BNDJ + theta
col. coming off a bound
this is new col value
Mark JPØS
basic and
unmark JOUT
JOUT goes
to a bound
Yes
Mark JOUT
at bound
if any rejected
cols clear reject
records
C - step solution
STEP
Beta + theta
x ALPHA
if pivot or
key change
pivot value = epsi
STATUS
display primal
step data
beginning of loop
B
End ITRN loop


END 1
Call ØPT1
IPHASE = 2
end of phase 1, ØPT1 displays STATUS
Go to A
END 2
Call ØPT2
end of phase 2, ØPT2 displays STATUS
RETURN
to LP with solution

Subroutine STATUS

STATUS prints BRIMAL data



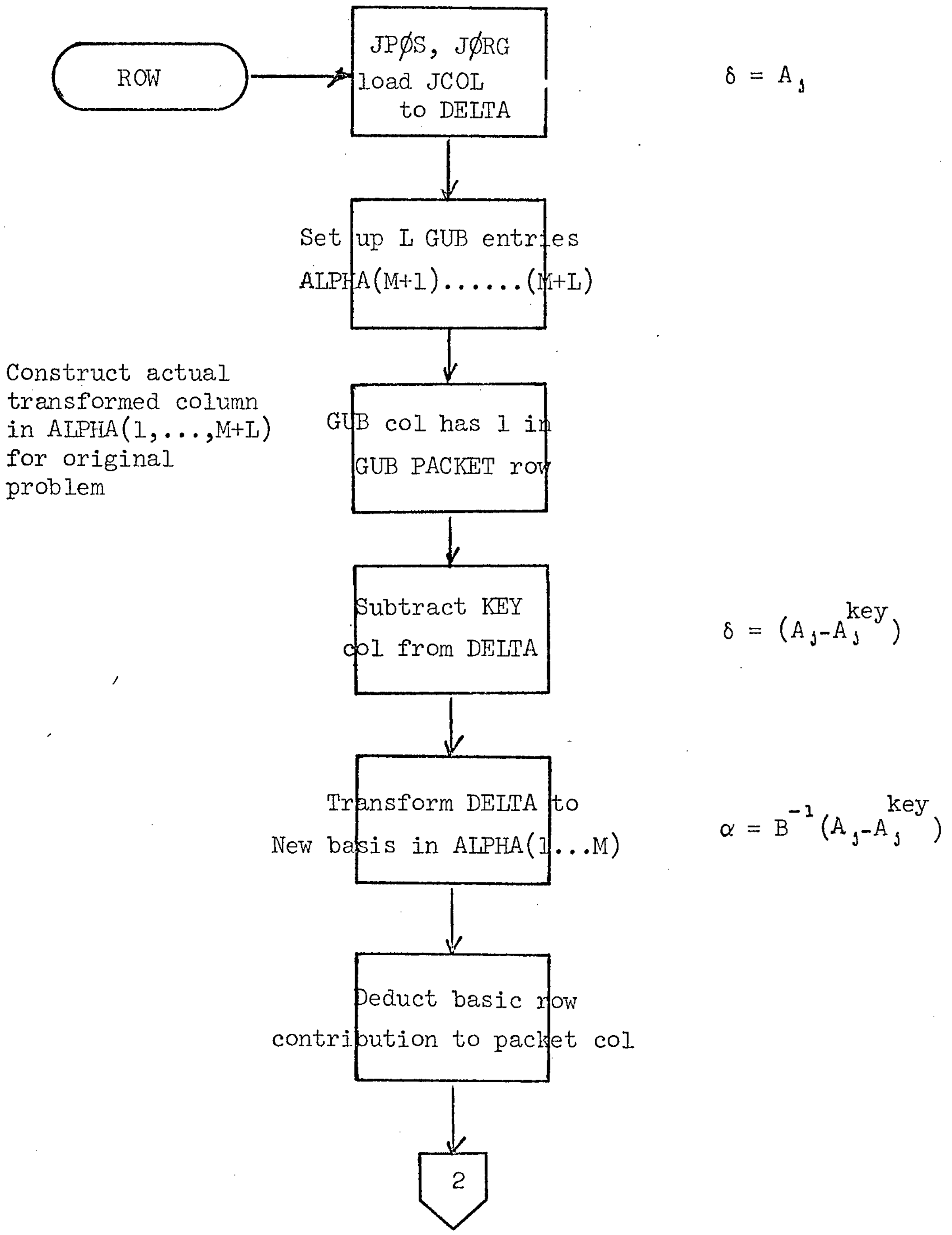
STATUS 2

ERRØR - triple prints error messages

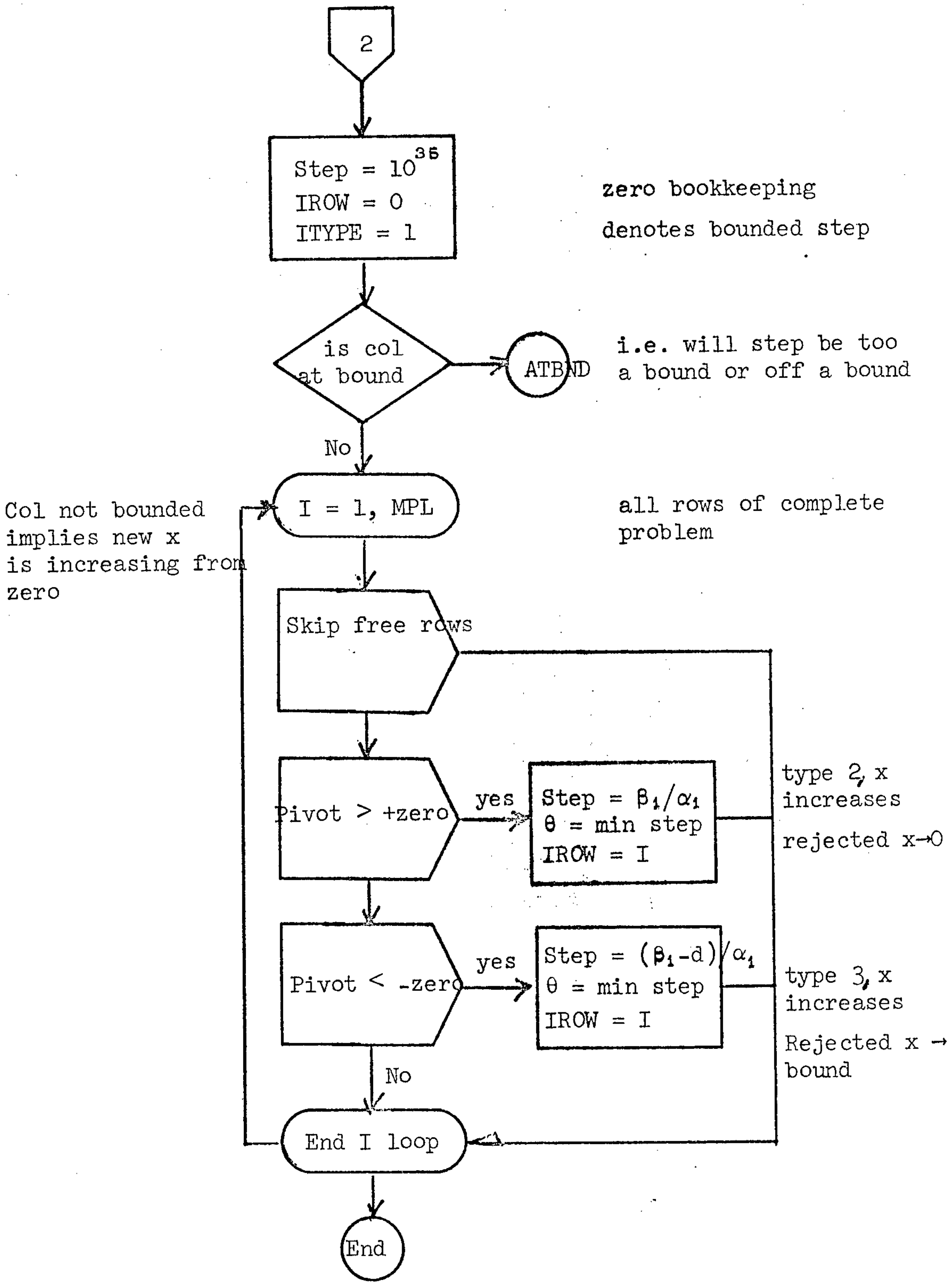
MESSG } single print error messages and print time in seconds since
MSSG } start of run.

Subroutine ROW

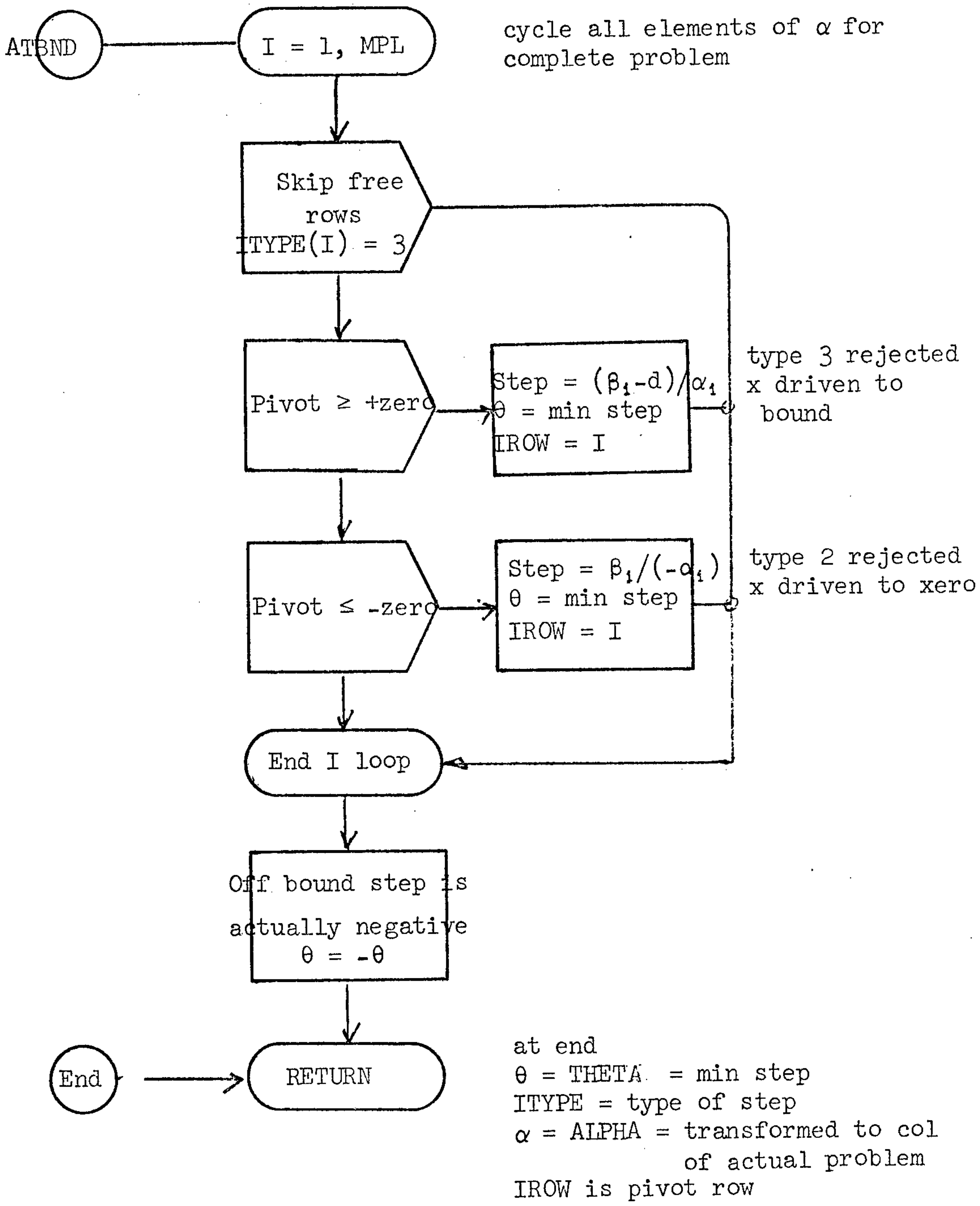
ROW computes current representation of selected column JCOL in core ALPHA then finds step MAX THETA which preserves feasibility.


ROW
JPØS, JØRG load JCOL to DELTA
$\delta = A_j$
Set up L GUB entries ALPHA(M+1)......(M+L)
Construct actual transformed column in ALPHA(1,...,M+L) for original problem
GUB col has 1 in GUB PACKET row
Subtract KEY col from DELTA
$\delta = (A_j - A_j^{key})$
Transform DELTA to New basis in ALPHA(1...M)
$\alpha = B^{-1}(A_j - A_j^{key})$
Deduct basic row contribution to packet col
2

ROW 2.


2
Step = 10^{35}
IROW = 0
ITYPE = 1
zero bookkeeping
denotes bounded step
is col at bound
ATBND
i.e. will step be too a bound or off a bound
No
I = 1, MPL
Col not bounded implies new x is increasing from zero
all rows of complete problem
Skip free rows
Pivot > +zero
yes
Step = β_i/α_i
θ = min step
IROW = I
type 2, x increases
rejected x→0
Pivot < -zero
yes
Step = $(\beta_i-d)/\alpha_i$
θ = min step
IROW = I
type 3, x increases
Rejected x → bound
No
End I loop
End

ROW 3.


ATBND
I = 1, MPL
cycle all elements of α for complete problem
Skip free rows ITYPE(I) = 3
Pivot ≥ +zero
Step = $(\beta_i - d)/\alpha_i$
θ = min step
IROW = I
type 3 rejected
x driven to bound
Pivot ≤ -zero
Step = $\beta_i/(-\alpha_i)$
θ = min step
IROW = I
type 2 rejected
x driven to xero
End I loop
Off bound step is actually negative
$\theta = -\theta$
End
RETURN
at end
θ = THETA = min step
ITYPE = type of step
α = ALPHA = transformed to col of actual problem
IROW is pivot row

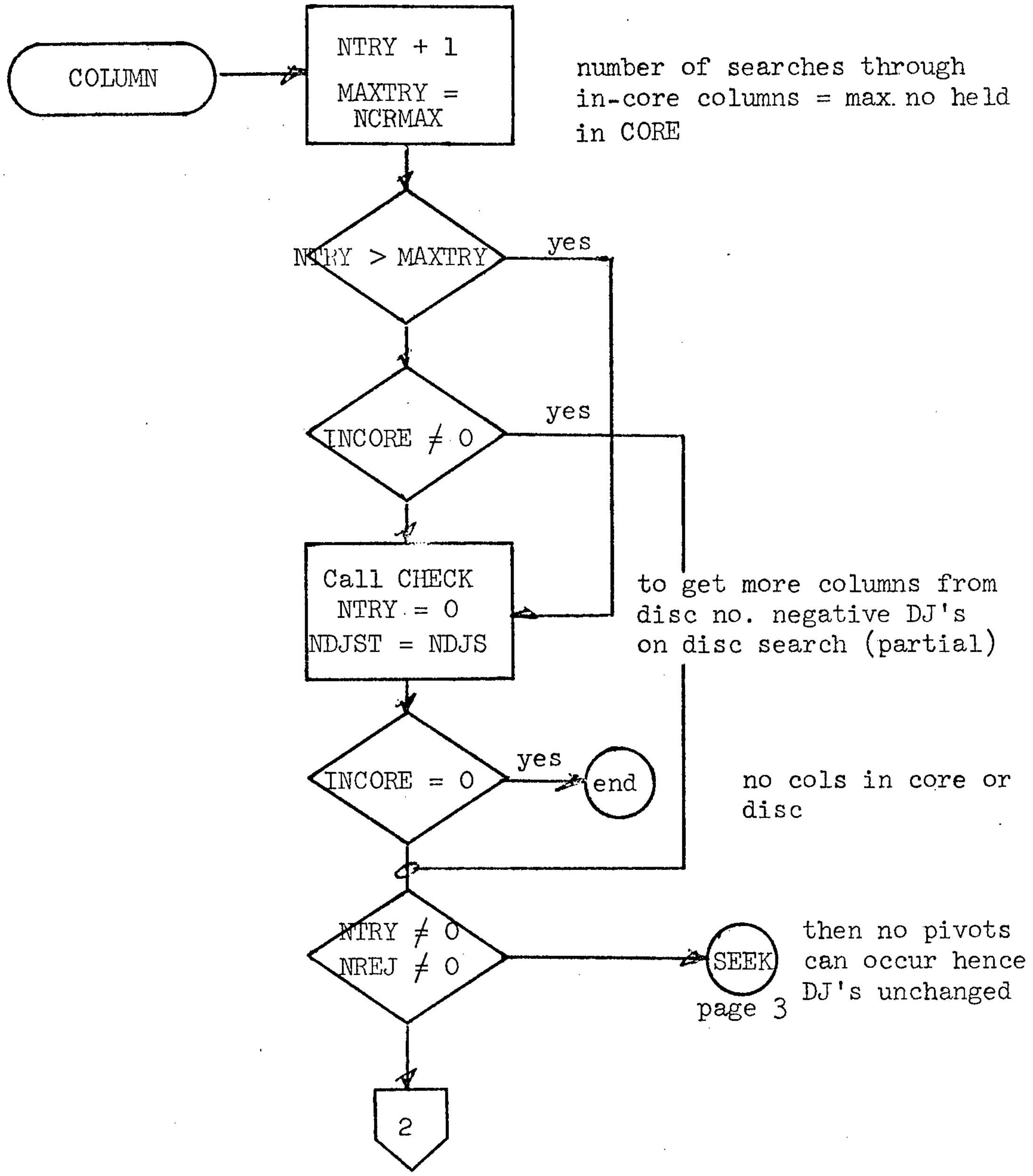
N.B. IROW ≤ M ⟹ pivot non-GUB row

> M ⟹ pivot on GUB row

Subroutine COLUMN

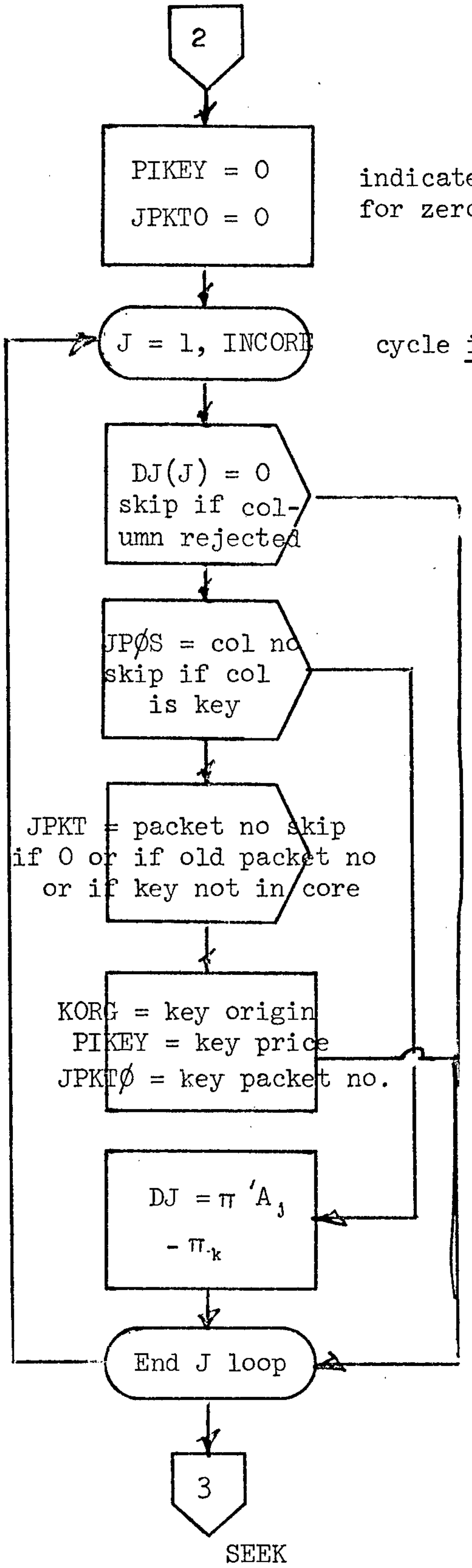
COLUMN selects a column JCOL from among vectors in core in AJ space.

If no columns price out, it calls CHECK to search disc for replenishment.


COLUMN
NTRY + 1
MAXTRY = NCRMAX
number of searches through in-core columns = max. no held in CORE
NTRY > MAXTRY
yes
INCORE ≠ 0
yes
Call CHECK
NTRY = 0
NDJST = NDJS
to get more columns from disc no. negative DJ's on disc search (partial)
INCORE = 0
yes
end
no cols in core or disc
NTRY ≠ 0
NREJ ≠ 0
SEEK
page 3
then no pivots can occur hence DJ's unchanged
2

COLUMN 2.

Reprices old columns in core unless the prices obviously haven't changed (same β^{-1}).



2
PIKEY = 0
JPKTO = 0
J = 1, INCORE
DJ(J) = 0 skip if column rejected
JPØS = col no skip if col is key
JPKT = packet no skip if 0 or if old packet no or if key not in core
KORG = key origin
PIKEY = key price
JPKTØ = key packet no.
DJ = π 'A_J - π_k
End J loop
3
SEEK

indicates no key price held for zero packet (old packet)

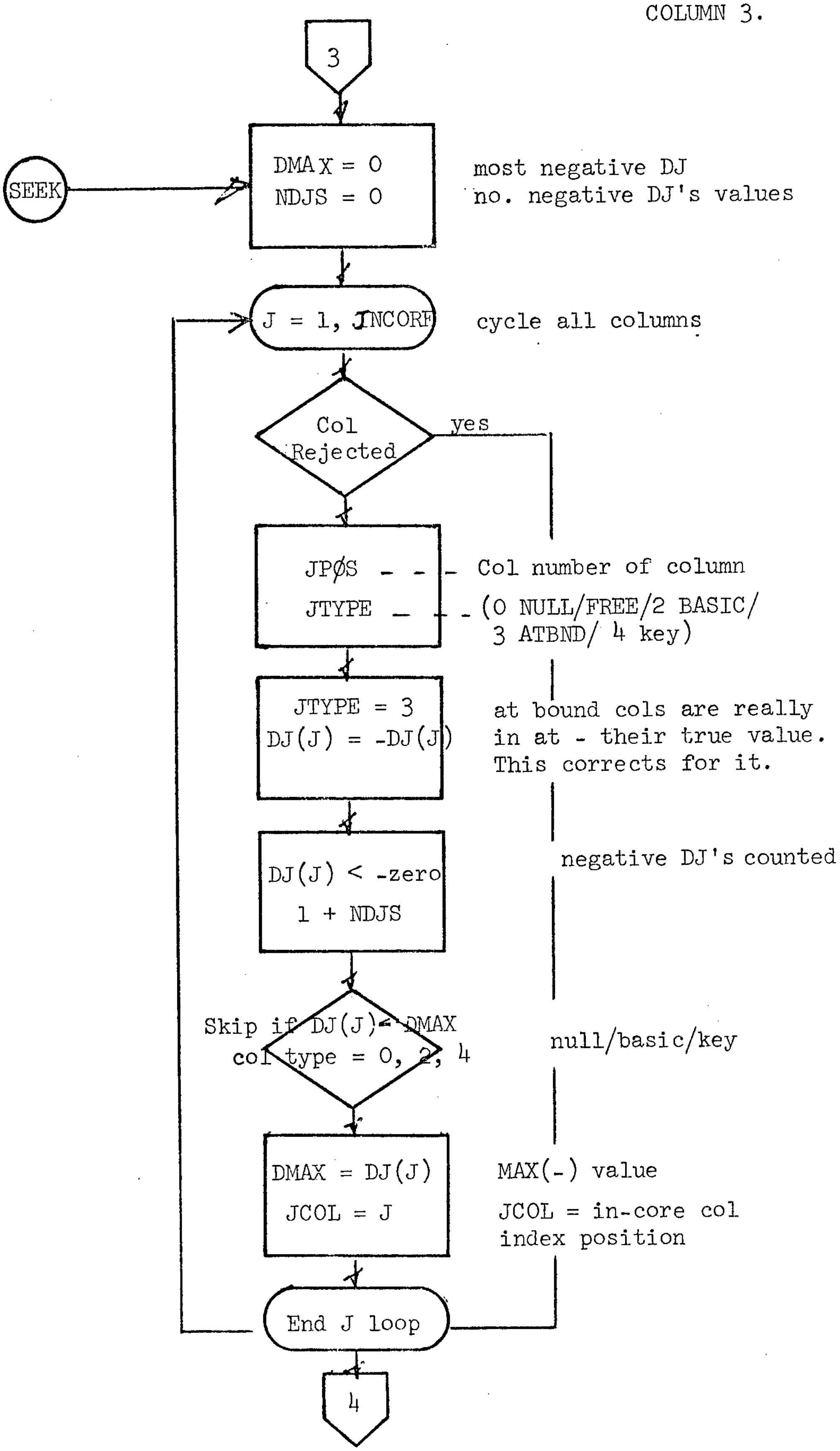
cycle in-core columns

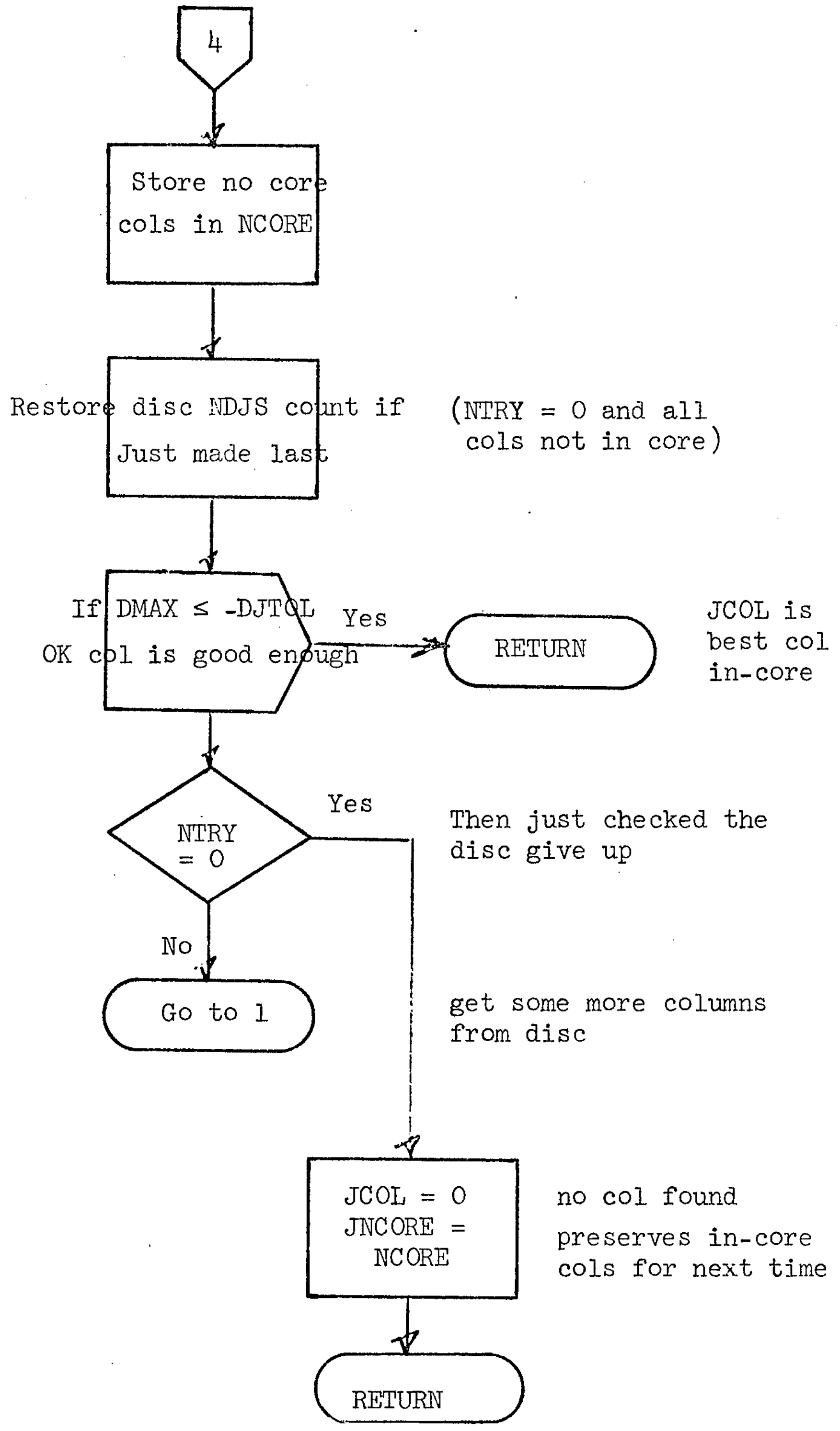
JPØS file col. no on disc

new packet, recompute price and store packet no.

price for GUB problem if non GUB, π_k = PIKEY ≡ 0
π is IC row of B^{-1}

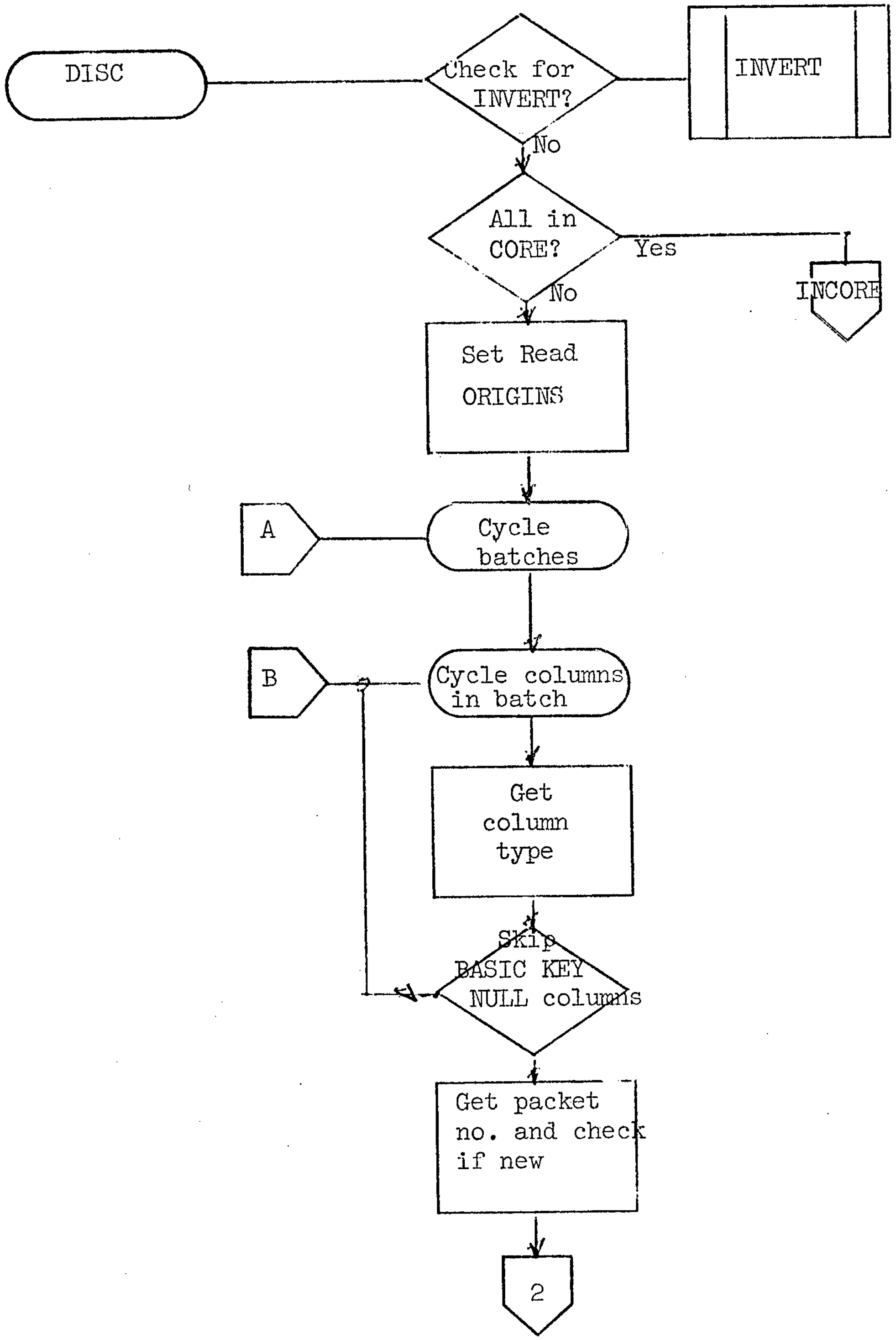
COLUMN 3.

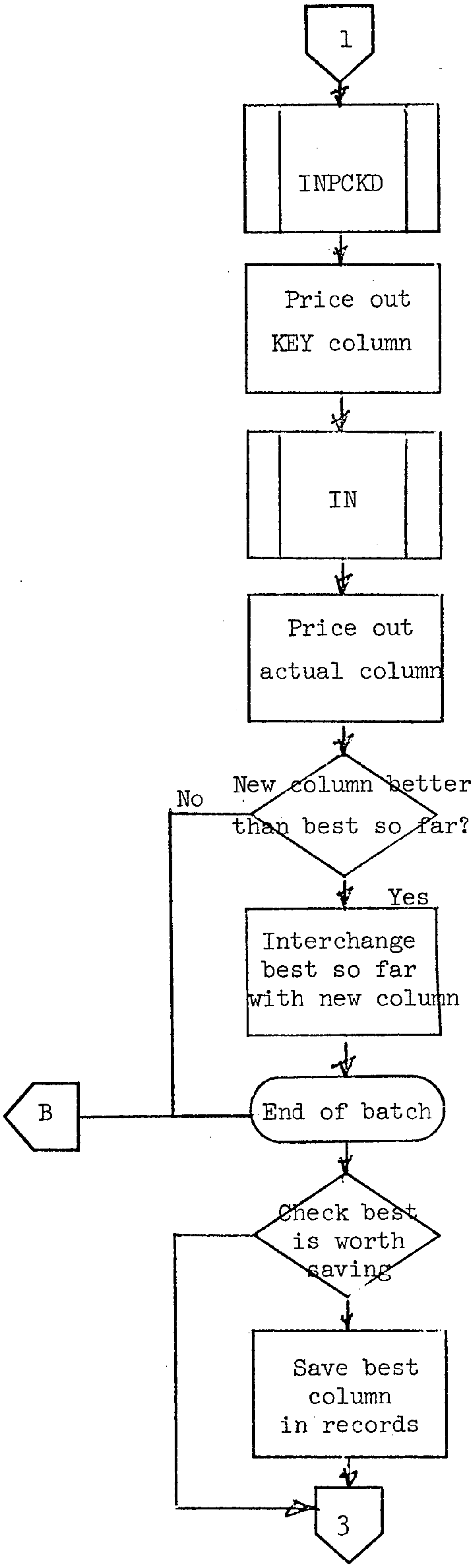

3
SEEK
DMAX = 0
NDJS = 0
most negative DJ
no. negative DJ's values
J = 1, JNCORE
cycle all columns
Col
Rejected
yes
JPØS
JTYPE
Col number of column
(0 NULL/FREE/2 BASIC/
3 ATBND/ 4 key)
JTYPE = 3
DJ(J) = -DJ(J)
at bound cols are really
in at - their true value.
This corrects for it.
DJ(J) < -zero
1 + NDJS
negative DJ's counted
Skip if DJ(J) < DMAX
col type = 0, 2, 4
null/basic/key
DMAX = DJ(J)
JCOL = J
MAX(-) value
JCOL = in-core col
index position
End J loop
4

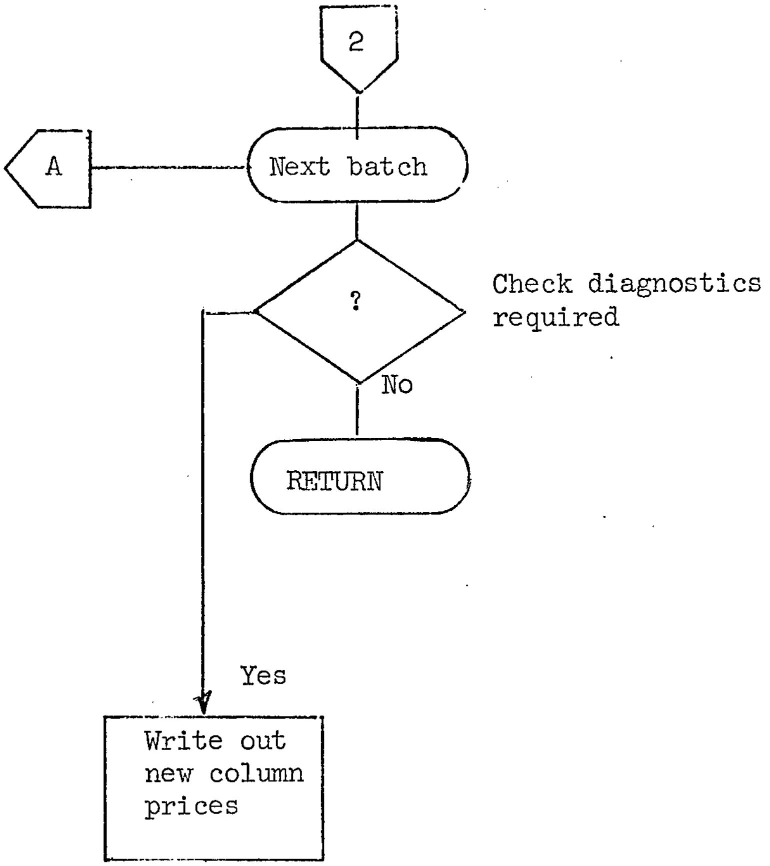

4
Store no core cols in NCORE
Restore disc NDJS count if Just made last
(NTRY = 0 and all cols not in core)
If DMAX ≤ -DJTOL OK col is good enough
Yes
RETURN
JCOL is best col in-core
NTRY = 0
Yes
Then just checked the disc give up
No
Go to 1
get some more columns from disc
JCOL = 0 JNCORE = NCORE
no col found
preserves in-core cols for next time
RETURN

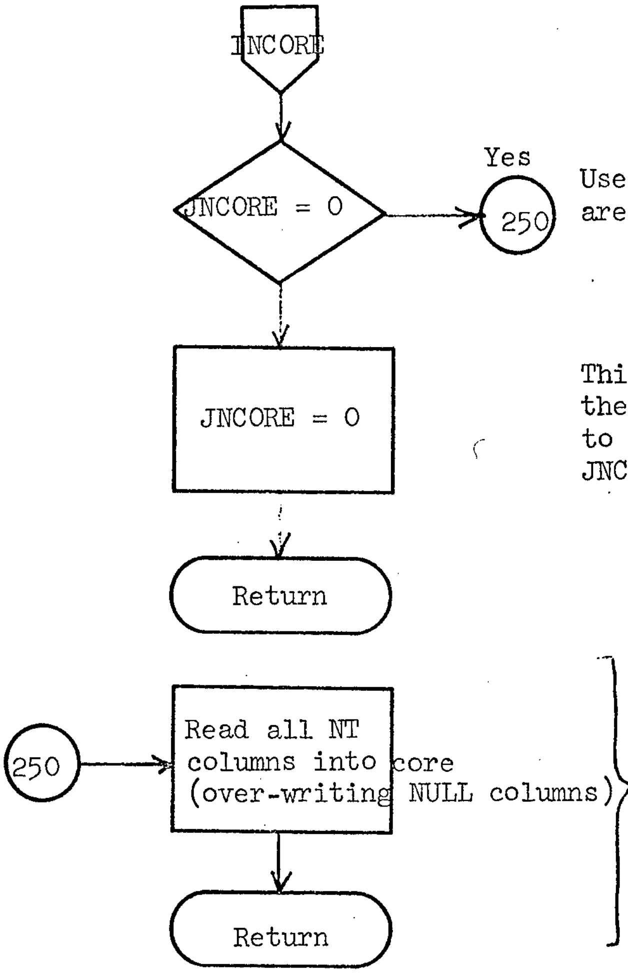
Subroutine DISC

Checks if an inverse is necessary and then checks the DISC files IA1, IA2 for more useful columns using IN and INPCKD.




1
INPCKD
Call in KEY column from packet file to KORG
Price out KEY column
IN
Call in actual column to JORG
Price out actual column
New column better than best so far?
No
Yes
Interchange best so far with new column
B
End of batch
Check best is worth saving
Save best column in records
3


2
A
Next batch
?
Check diagnostics required
No
RETURN
Yes
Write out new column prices



Used initially when there are no columns in core

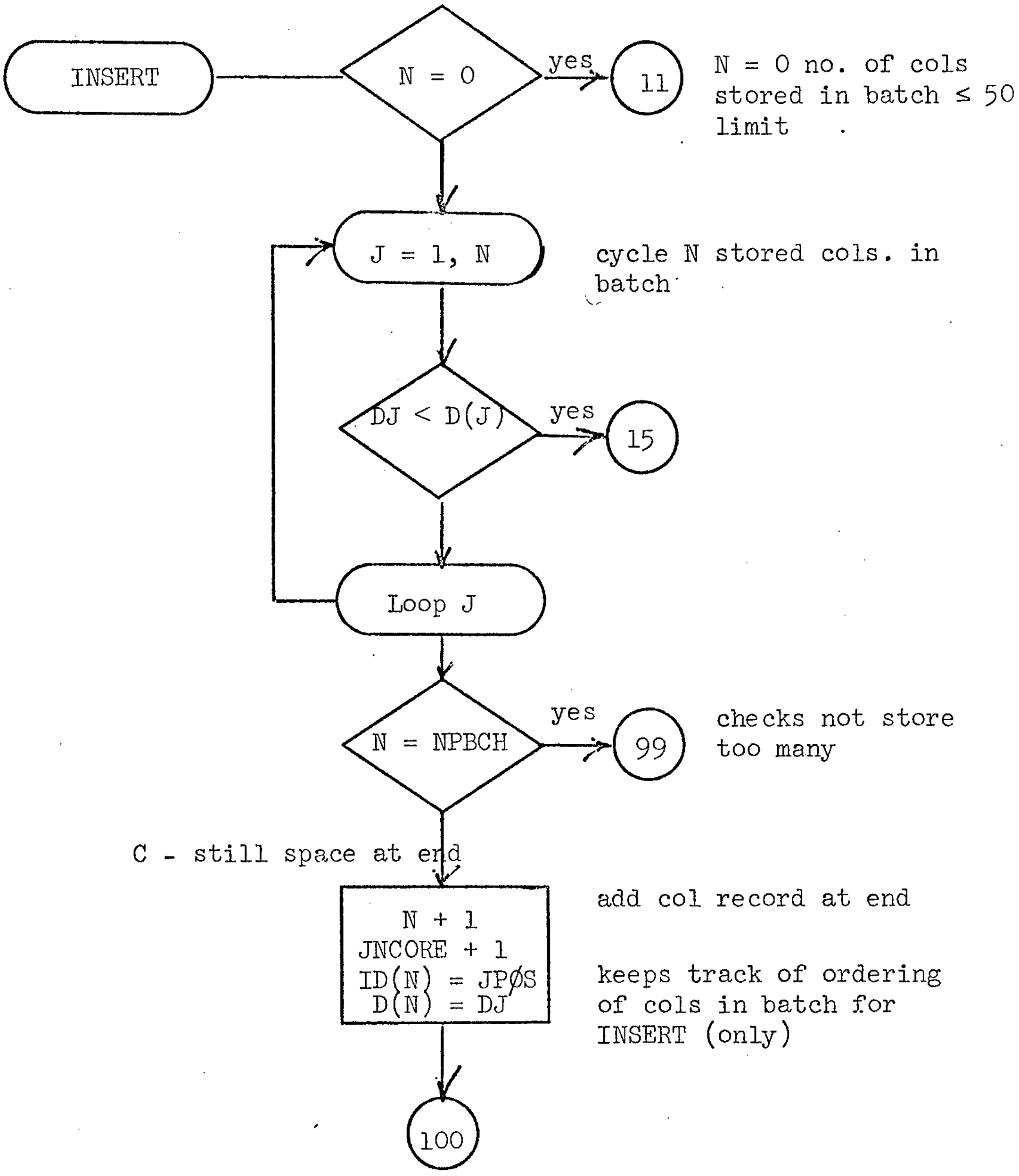
This indicates that there are no other cols to COLUMN which restores JNCORE to NT.

Places all NT cols into core and sets JNCORE to column count

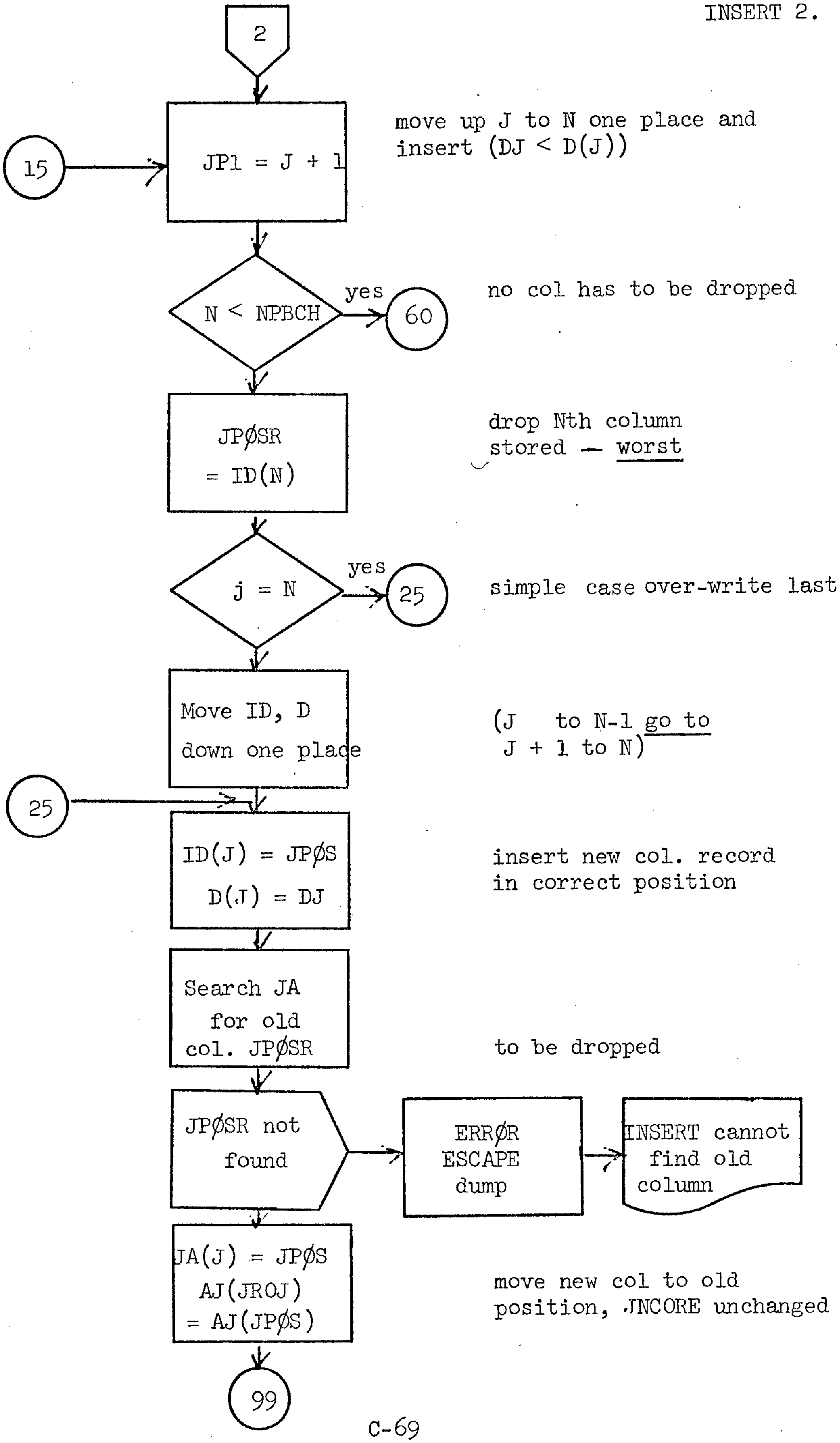
N.B. Once the columns are read into core, they stay there because COLUMN always restores JNCORE to NT and keeps track of them at the end of each phase.

Subroutine INSERT INSERT 1.

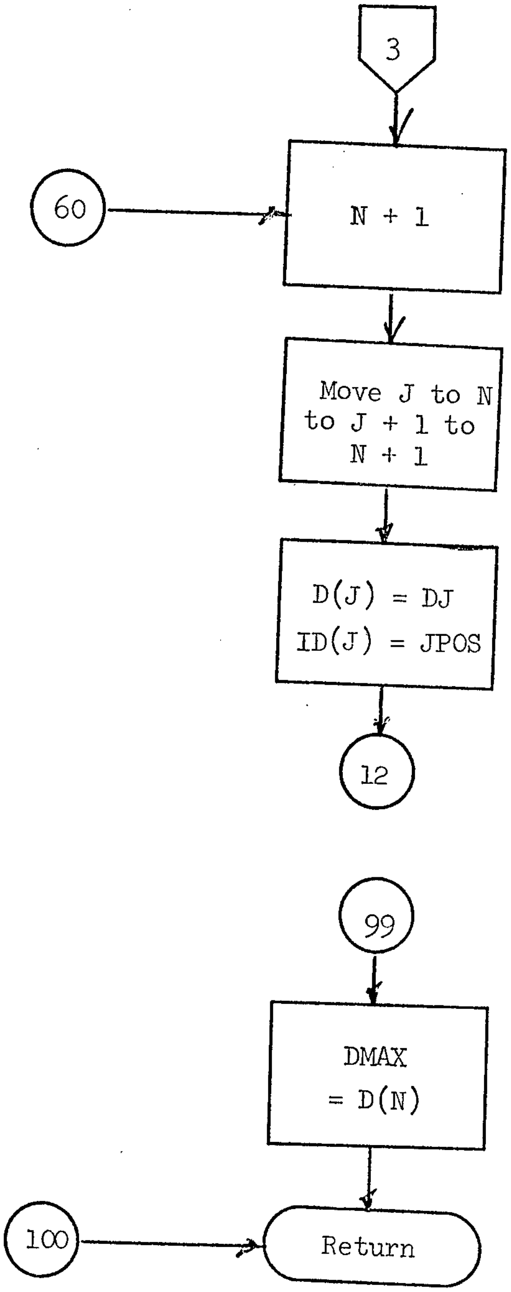
INSERT used by CHECK to order columns selected in the batch. It keeps track of worst column stored of the batch and over writes it if a better one is offered.


INSERT
N = 0
yes
11
N = 0 no. of cols stored in batch ≤ 50 limit
J = 1, N
cycle N stored cols. in batch
DJ < D(J)
yes
15
Loop J
N = NPBCH
yes
99
checks not store too many
C - still space at end
N + 1
JNCORE + 1
ID(N) = JPØS
D(N) = DJ
add col record at end
keeps track of ordering of cols in batch for INSERT (only)
100

INSERT 2.


2
15
JP1 = J + 1
move up J to N one place and
insert (DJ < D(J))
N < NPBCH
yes
60
no col has to be dropped
JPØSR
= ID(N)
drop Nth column
stored — worst
j = N
yes
25
simple case over-write last
Move ID, D
down one place
(J to N-1 go to
J + 1 to N)
25
ID(J) = JPØS
D(J) = DJ
insert new col. record
in correct position
Search JA
for old
col. JPØSR
to be dropped
JPØSR not
found
ERRØR
ESCAPE
dump
INSERT cannot
find old
column
JA(J) = JPØS
AJ(JROJ)
= AJ(JPØS)
move new col to old
position, JNCORE unchanged
99

INSERT 3.



move up J to N one place saving N

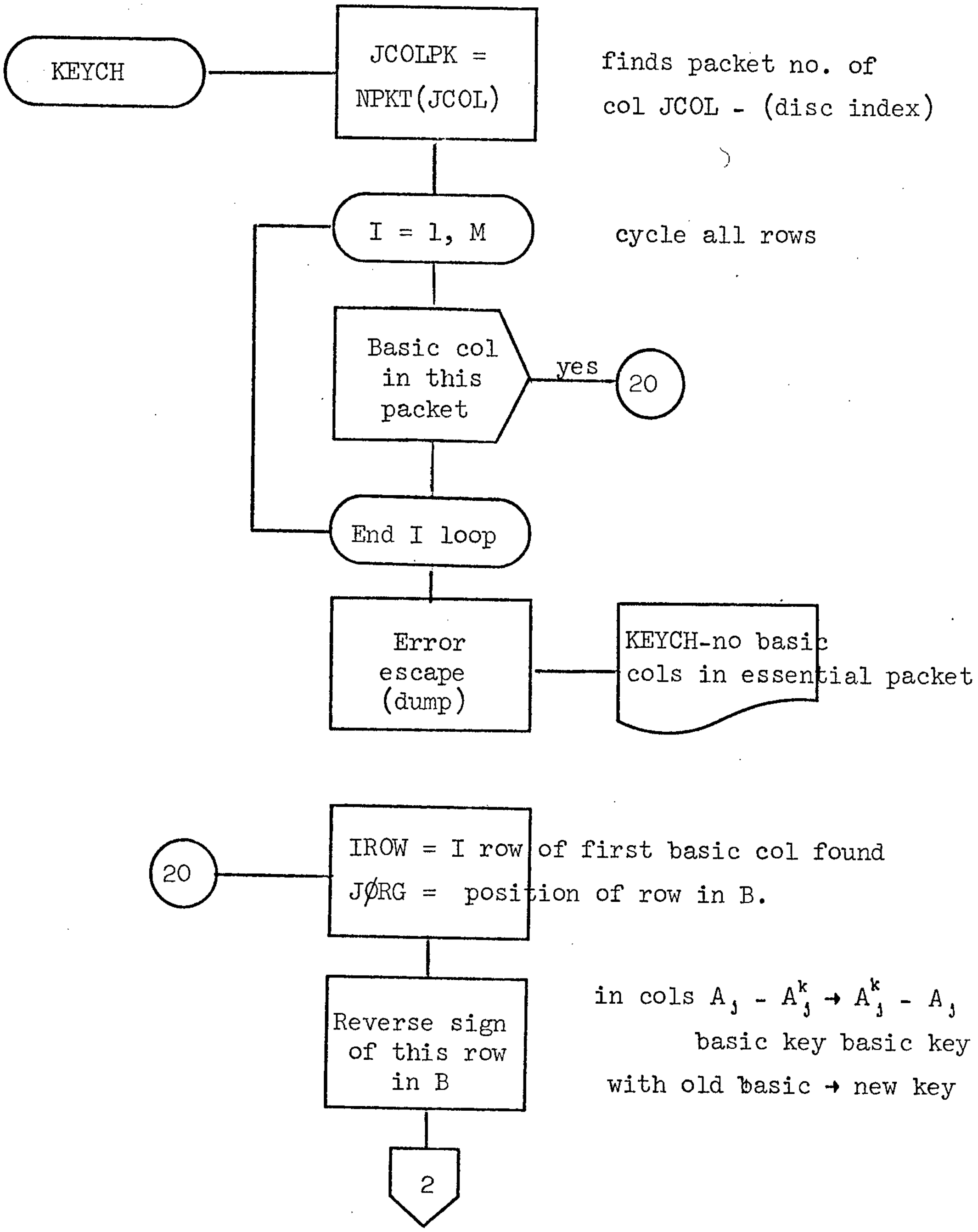
save Jth in correct place

go to move up records

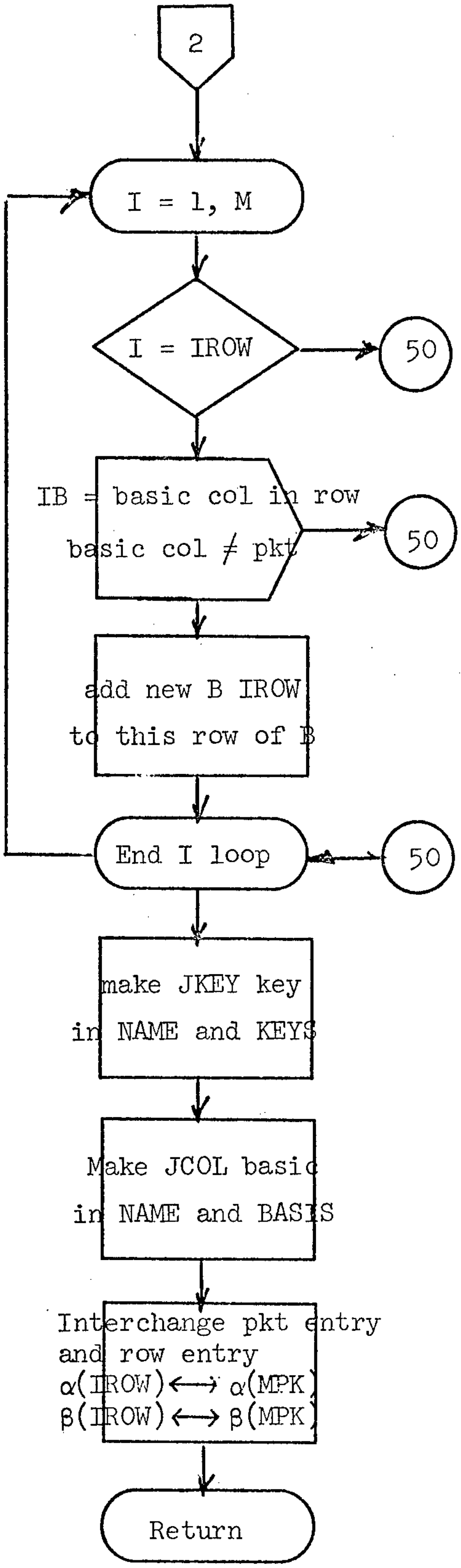
save value of largest D for use in comparing new cols in CHECK

Subroutine KEYCH

KEYCH changes key to make JCOL basic in some row IROW found, making old col key, and corrects α, β and B^{-1}.


KEYCH
JCOLPK = NPKT(JCOL)
finds packet no. of col JCOL - (disc index)
I = 1, M
cycle all rows
Basic col in this packet
yes
20
End I loop
Error escape (dump)
KEYCH-no basic cols in essential packet
20
IROW = I row of first basic col found
JØRG = position of row in B.
Reverse sign of this row in B
in cols $A_j - A_j^k \to A_j^k - A_j$
basic key basic key
with old basic → new key
2

KEYCH 2.



Rearrange B^{-1} in B for new key which was basic

skip IROW

check row is in same GUB packet

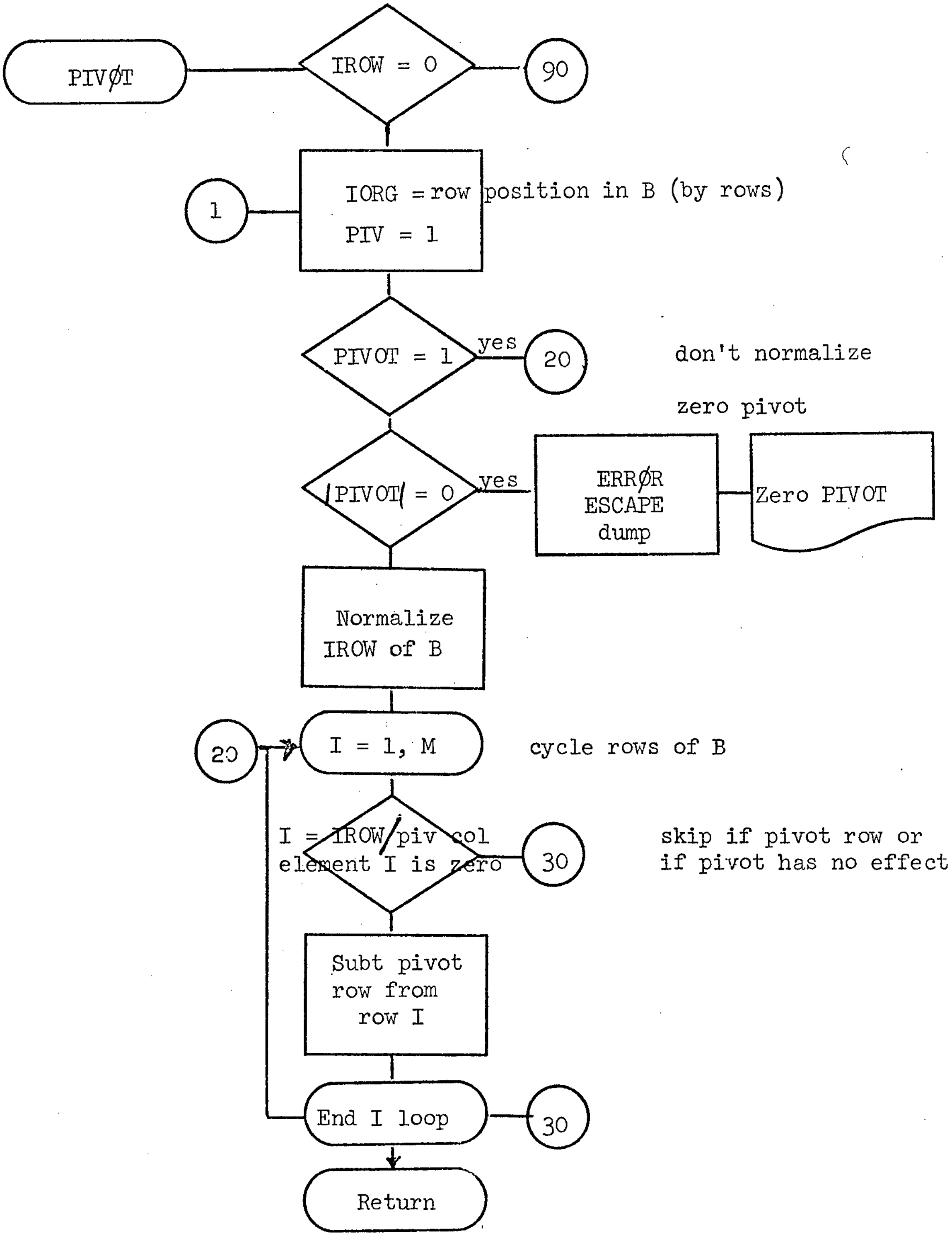
add B(IROW) to B(I) to get $A^I_j - A^k_j$ for new key

correct basis and key markers

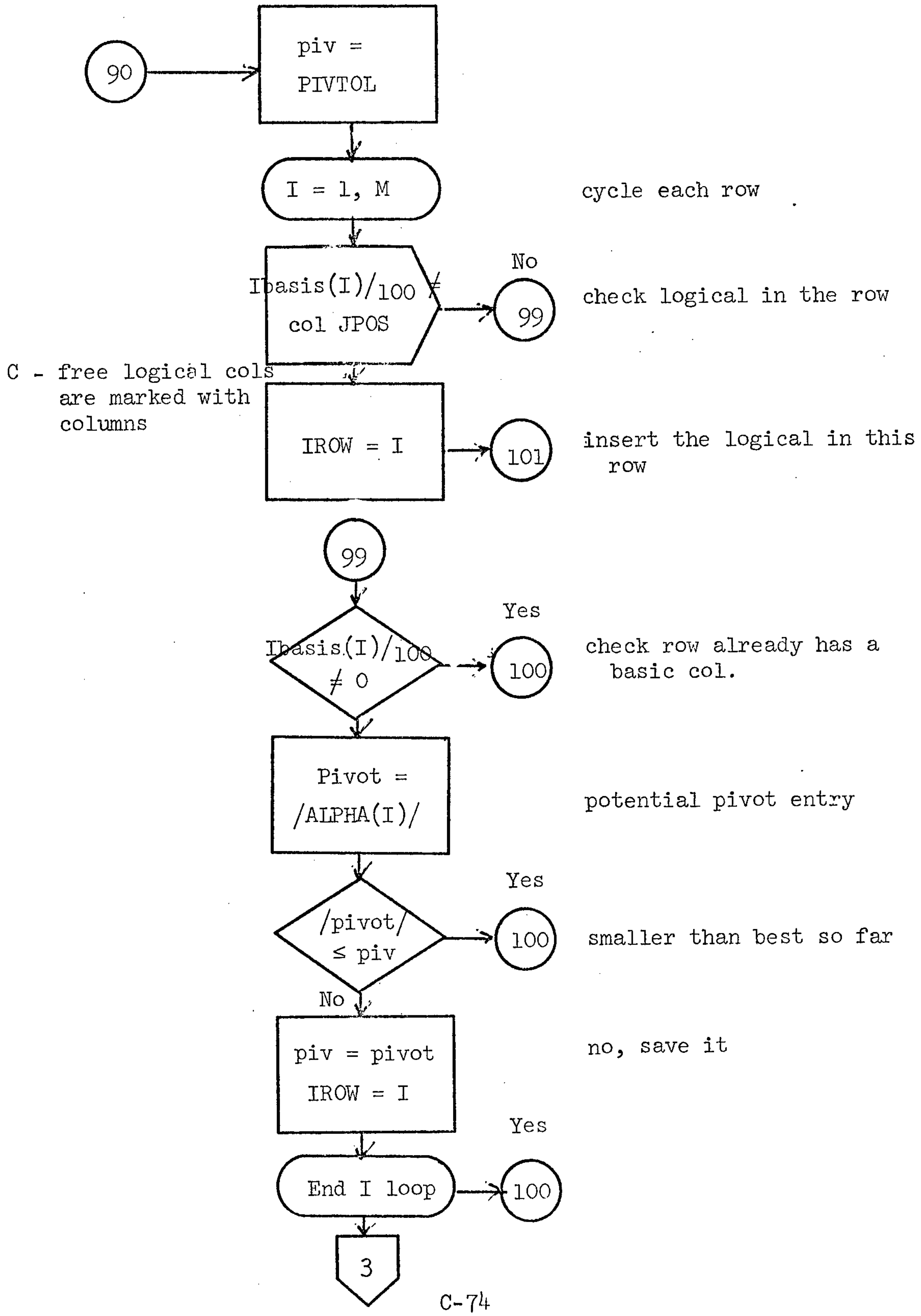
correct α, β for new representation

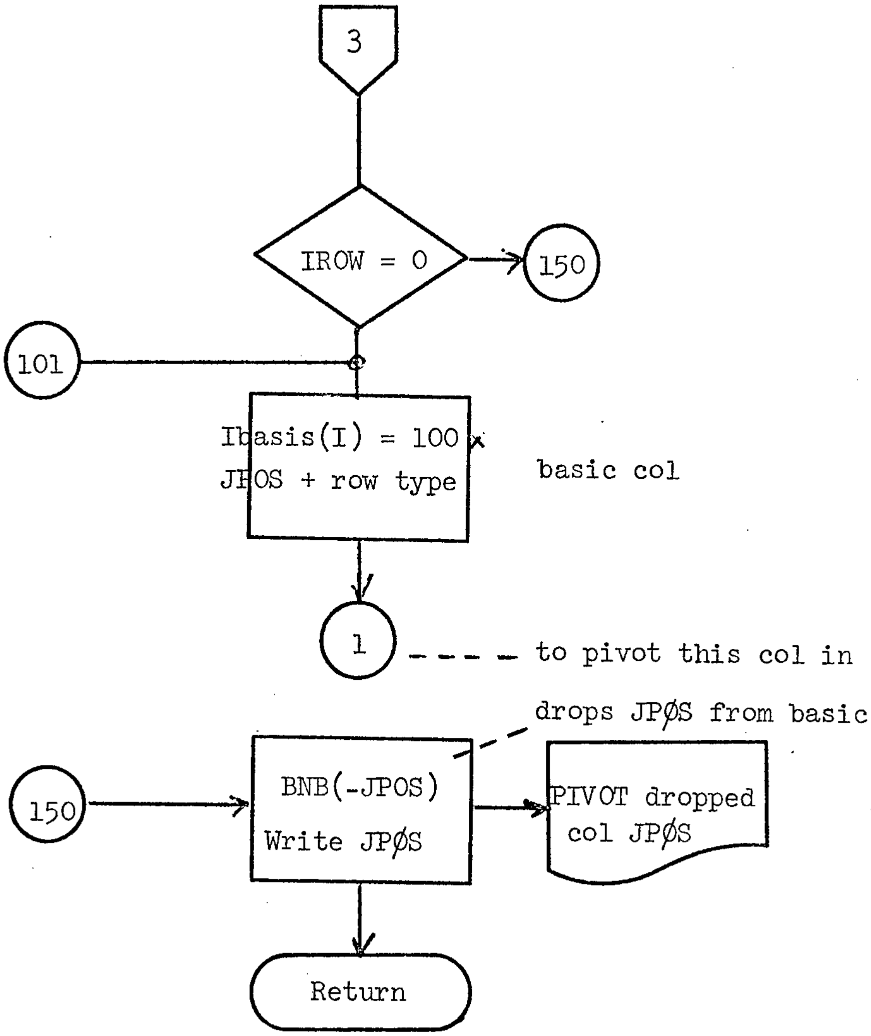
Subroutine PIVOT

PIVOT inserts current representation of column ALPHA into basic inverse B at IROW, if IROW = 0 it finds a slot for ALPHA or rejects it.

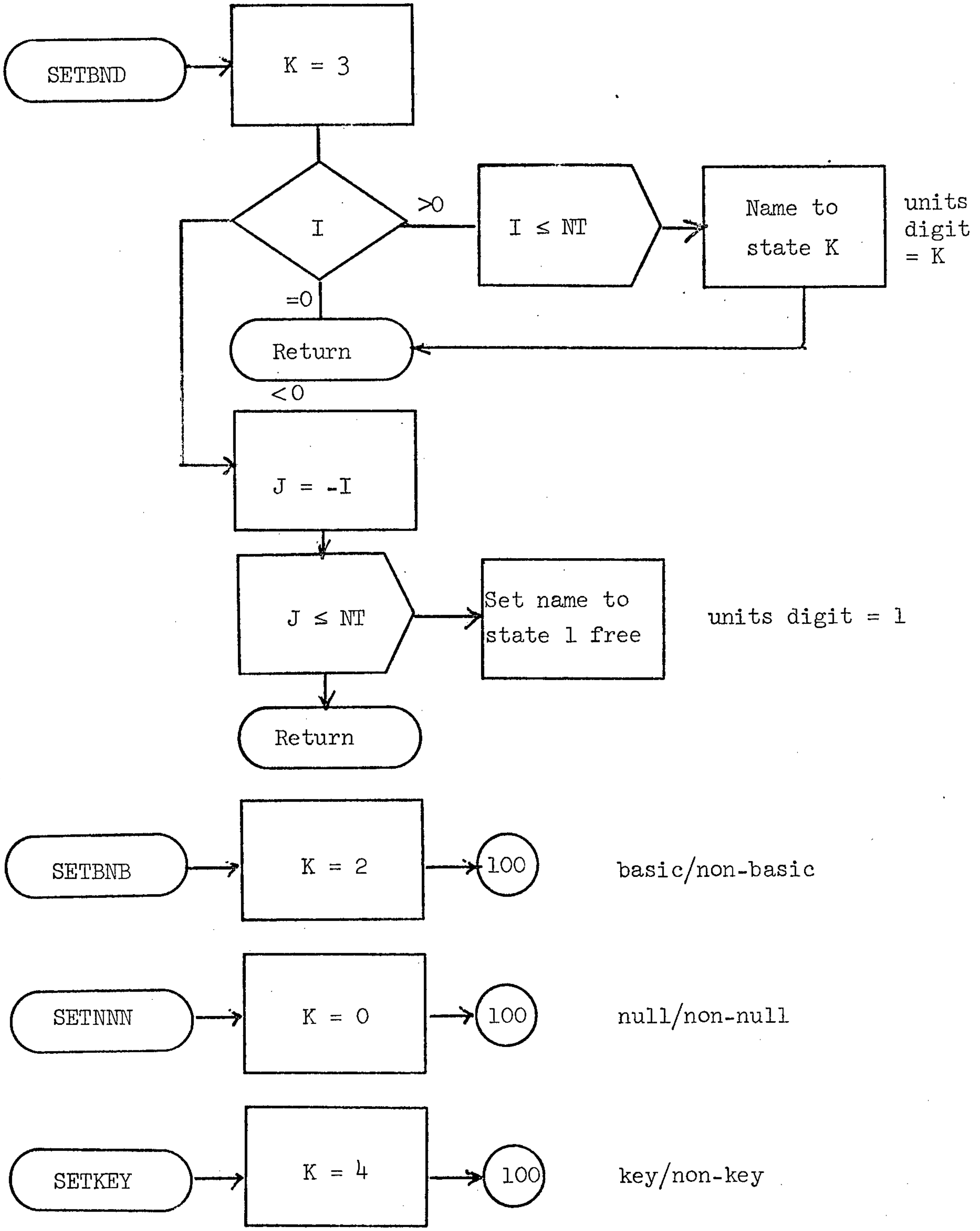


Find best row for ALPHA called from INVERT when row not known.




3
IROW = 0
150
101
Ibasis(I) = 100 x
JPOS + row type
basic col
1
to pivot this col in
drops JPØS from basic
150
BNB(-JPOS)
Write JPØS
PIVOT dropped
col JPØS
Return

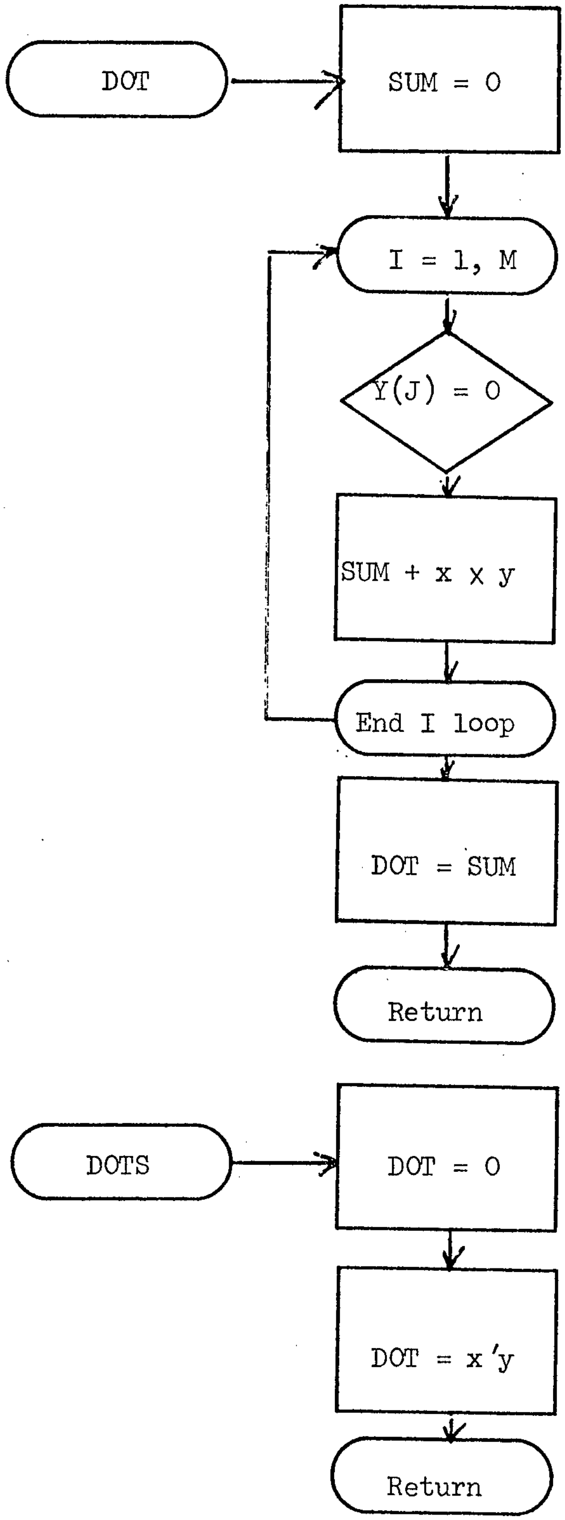
SETBND, SETBNB, SETKEY, SETNNN all set or unset to state of a variable to bound/basic/key/null.



DOT
DOTS

FUNCTION DOT, DOTS

DOT and DOTS evaluate double and single precision inner products DOT = x'y.



double precision
accumalator

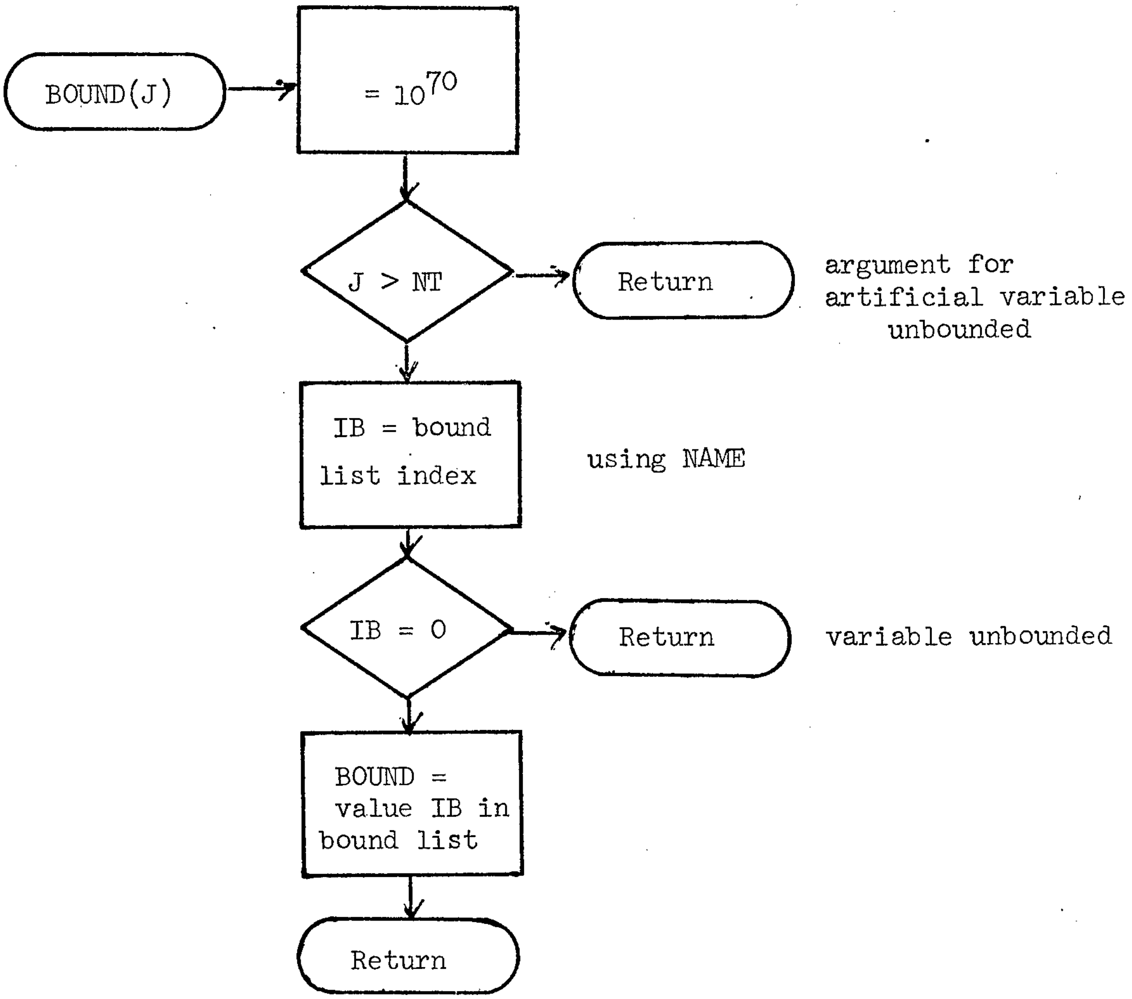
in x'y y elements
are often zero (as used)
avoid double precision
manipulations for this
case

Replace double answer
to single

quick single precision
version to save time

FUNCTION BOUND

BOUND - checks its argument and picks up the variable bound value.

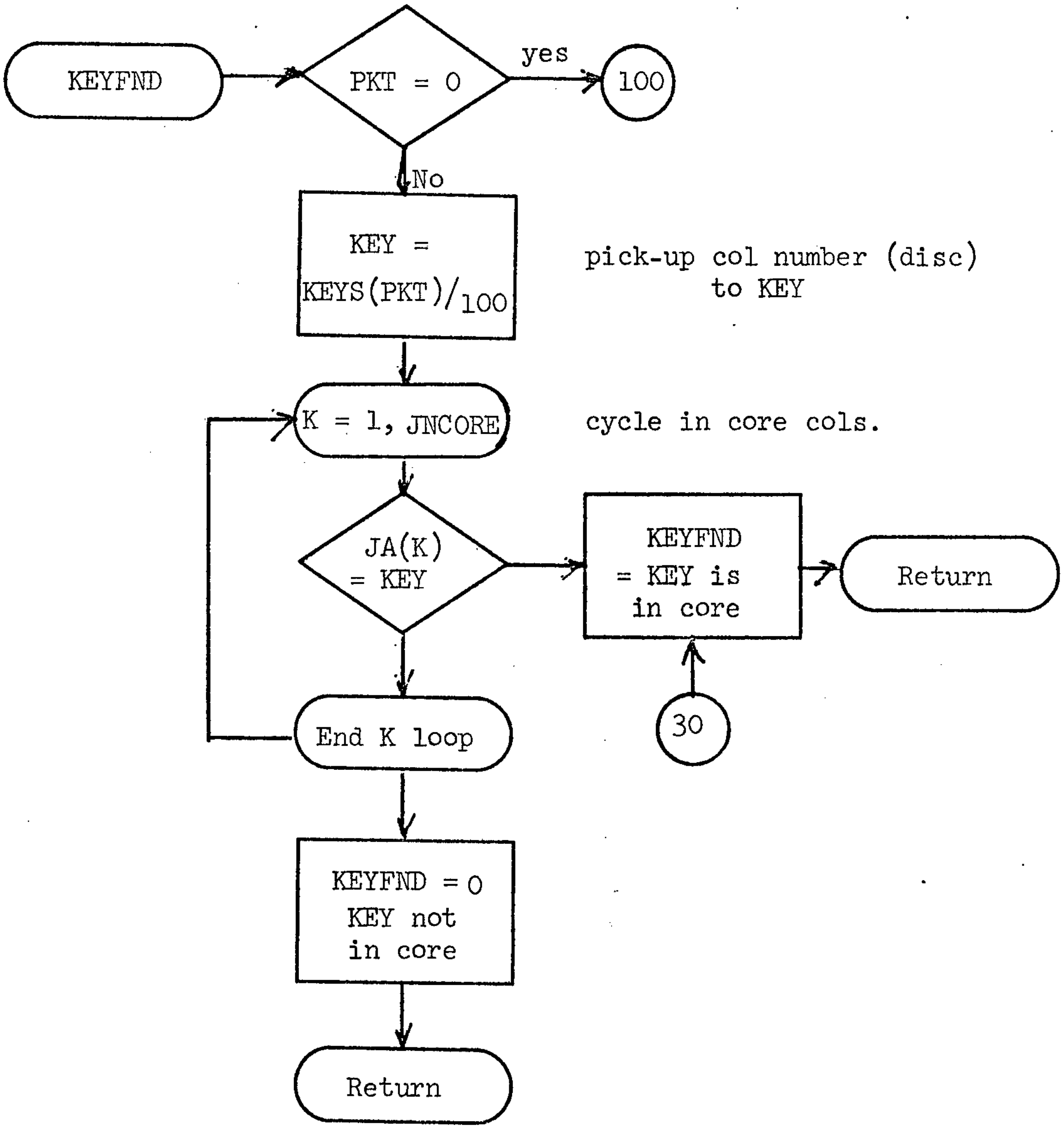


NB. NAME format has least significant decimal digits as shown.

```
.......0 0·B B|K K K K|S
              |       |
bound index   | GUB   | state
    or 0      | packet|
```

Function KEYFND

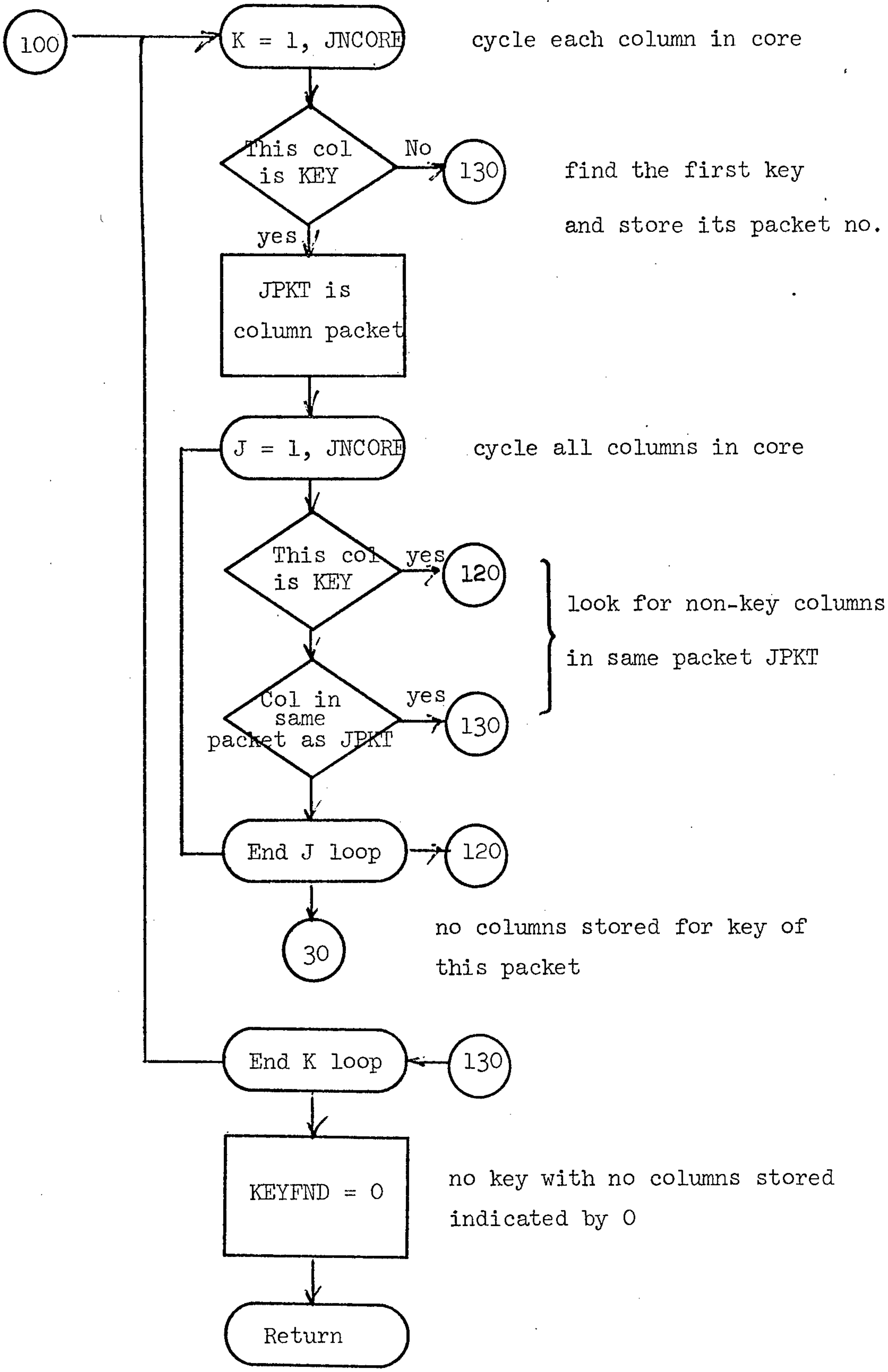
KEYFND finds key column of a packet in core or returns 0.



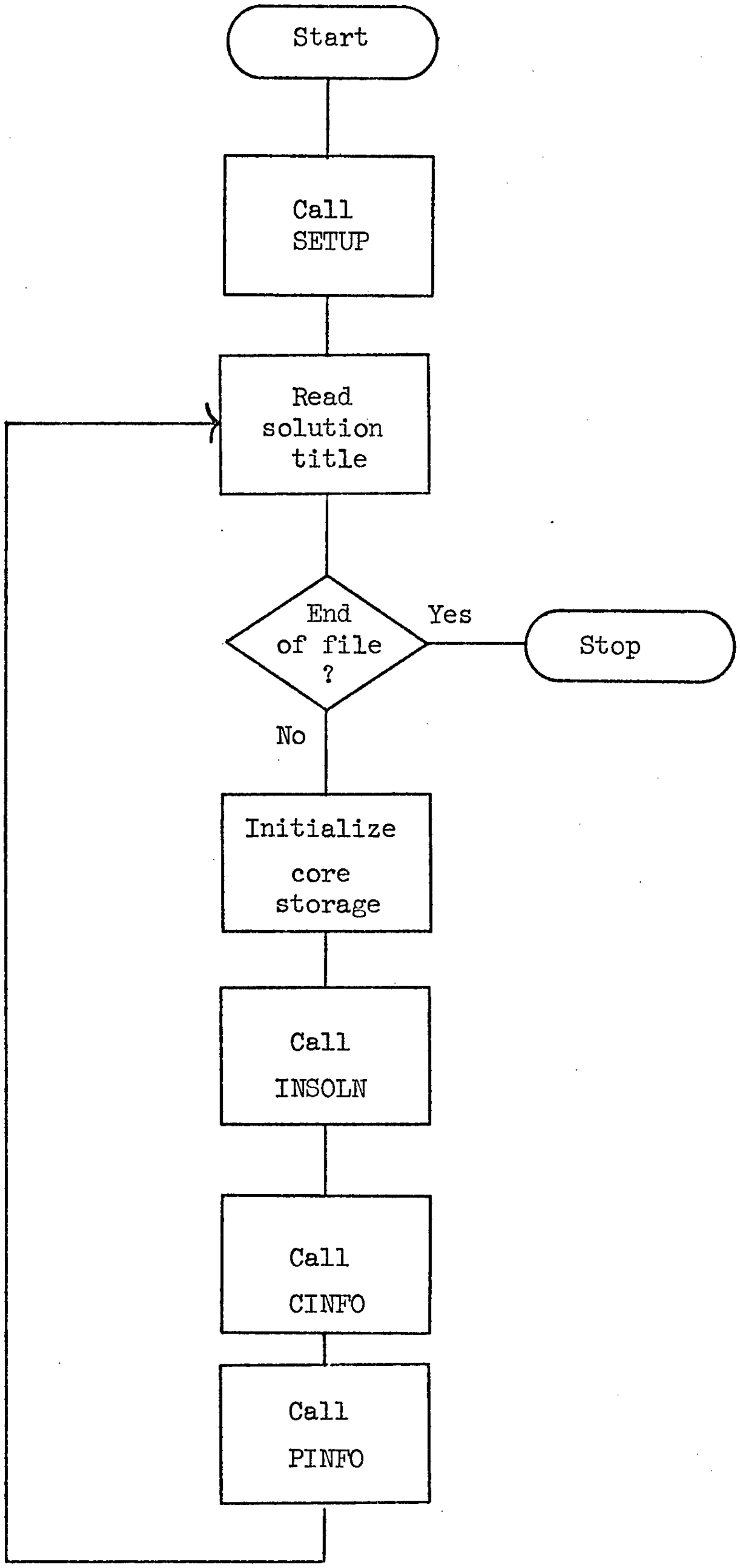
pick-up col number (disc) to KEY

cycle in core cols.

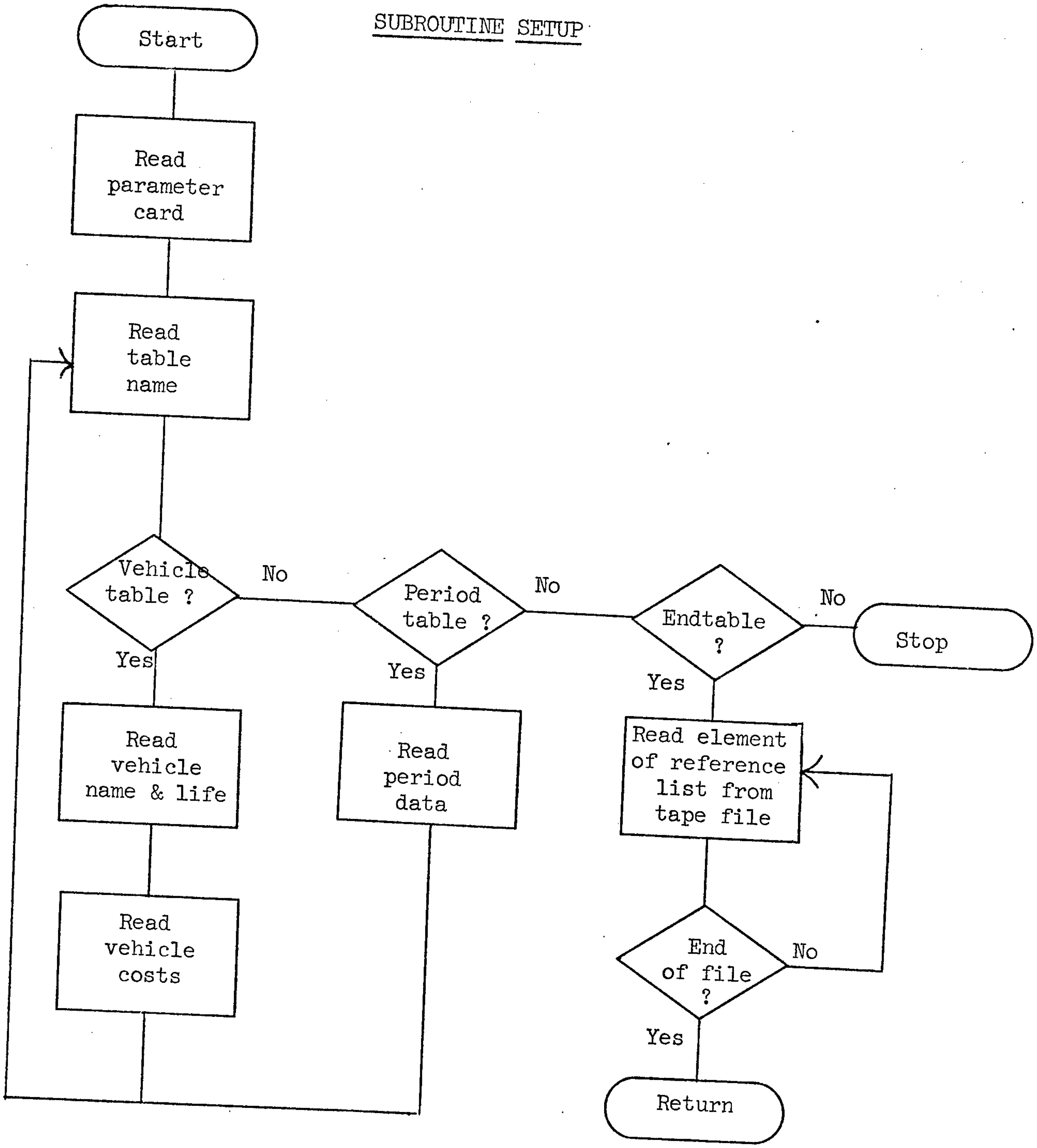
KEYFND 2.


100
K = 1, JNCORE
cycle each column in core
This col is KEY
No
130
find the first key
and store its packet no.
yes
JPKT is column packet
J = 1, JNCORE
cycle all columns in core
This col is KEY
yes
120
look for non-key columns
in same packet JPKT
Col in same packet as JPKT
yes
130
End J loop
120
30
no columns stored for key of
this packet
End K loop
130
KEYFND = 0
no key with no columns stored
indicated by 0
Return

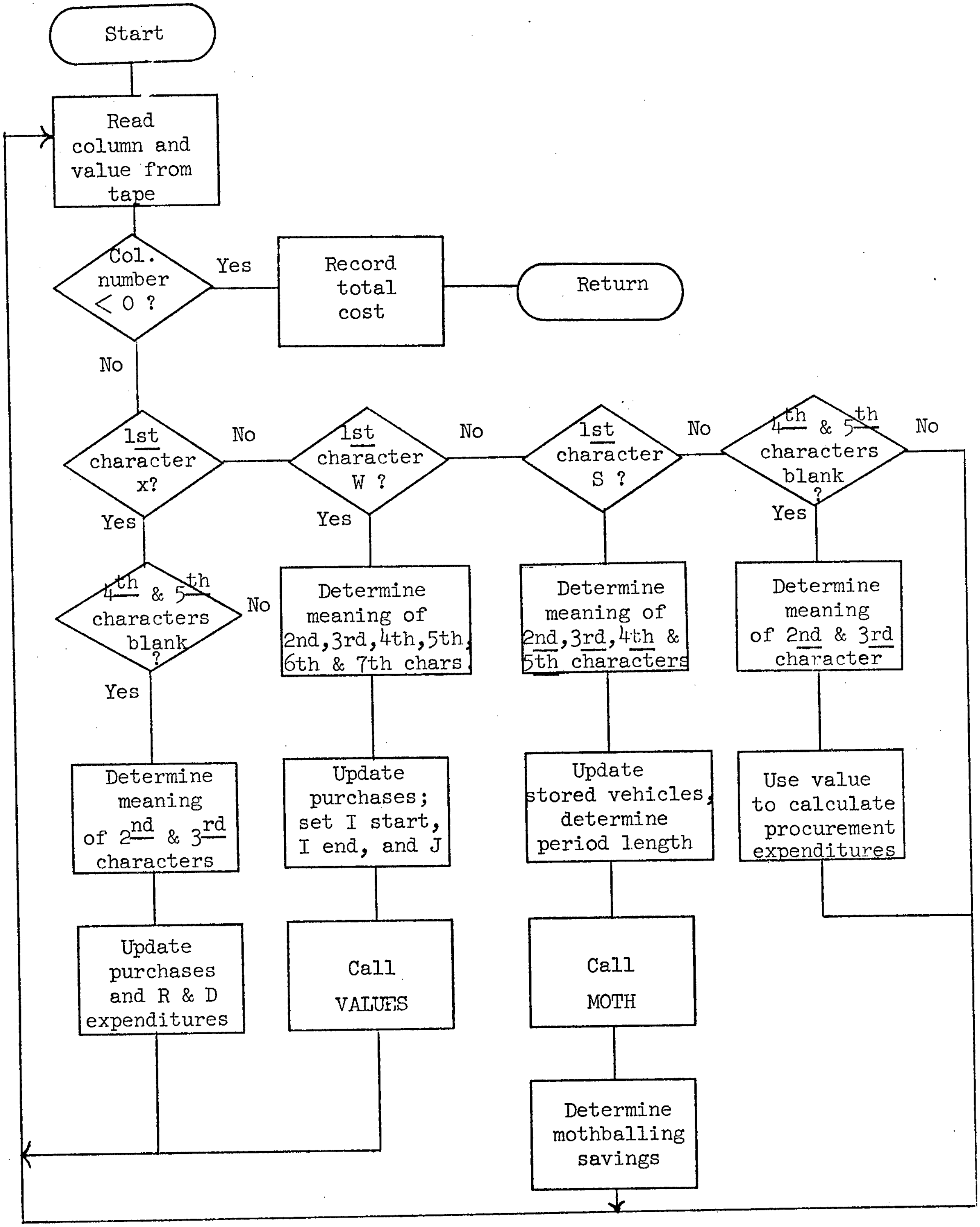
PROGRAM REPGEN



SUBROUTINE SETUP

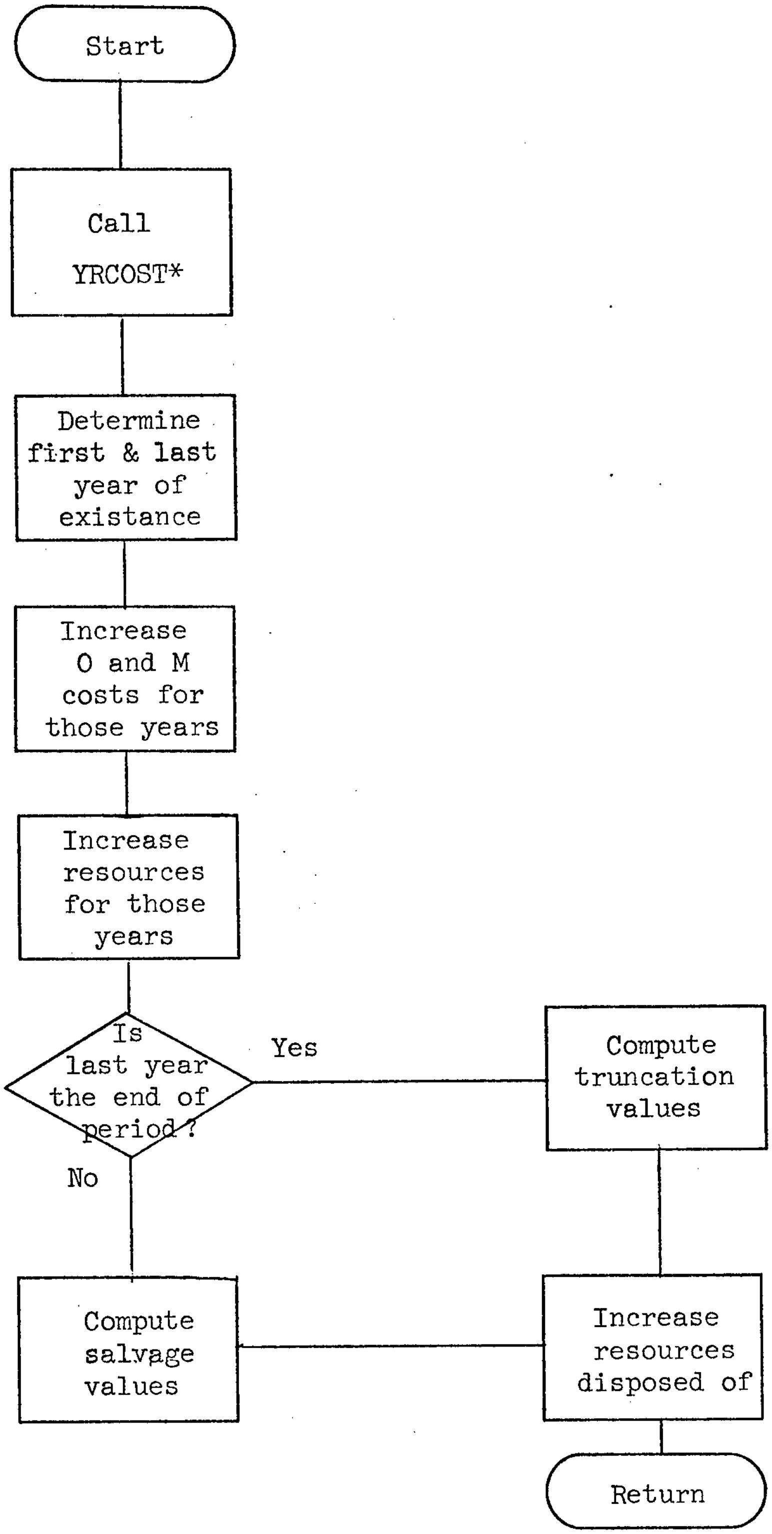


SUBROUTINE INSOLN



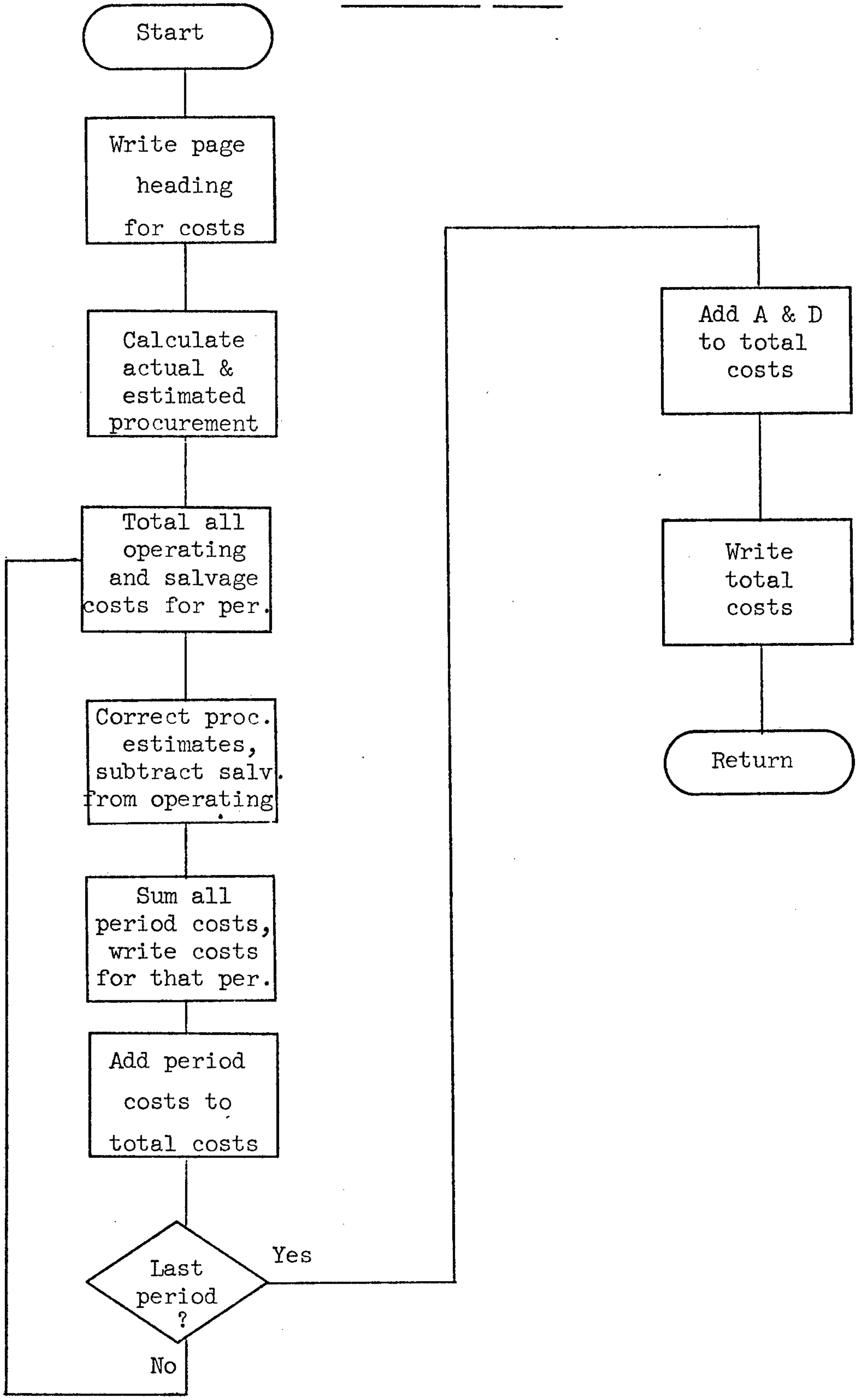
Start
Read column and value from tape
Col. number < 0 ?
Yes
Record total cost
Return
No
1st character x?
No
1st character W ?
No
1st character S ?
No
4th & 5th characters blank ?
No
Yes
4th & 5th characters blank ?
No
Yes
Determine meaning of 2nd,3rd,4th,5th, 6th & 7th chars
Yes
Determine meaning of 2nd,3rd,4th & 5th characters
Yes
Determine meaning of 2nd & 3rd character
Determine meaning of 2nd & 3rd characters
Update purchases; set I start, I end, and J
Update stored vehicles, determine period length
Use value to calculate procurement expenditures
Update purchases and R & D expenditures
Call VALUES
Call MOTH
Determine mothballing savings

SUBROUTINE VALUES

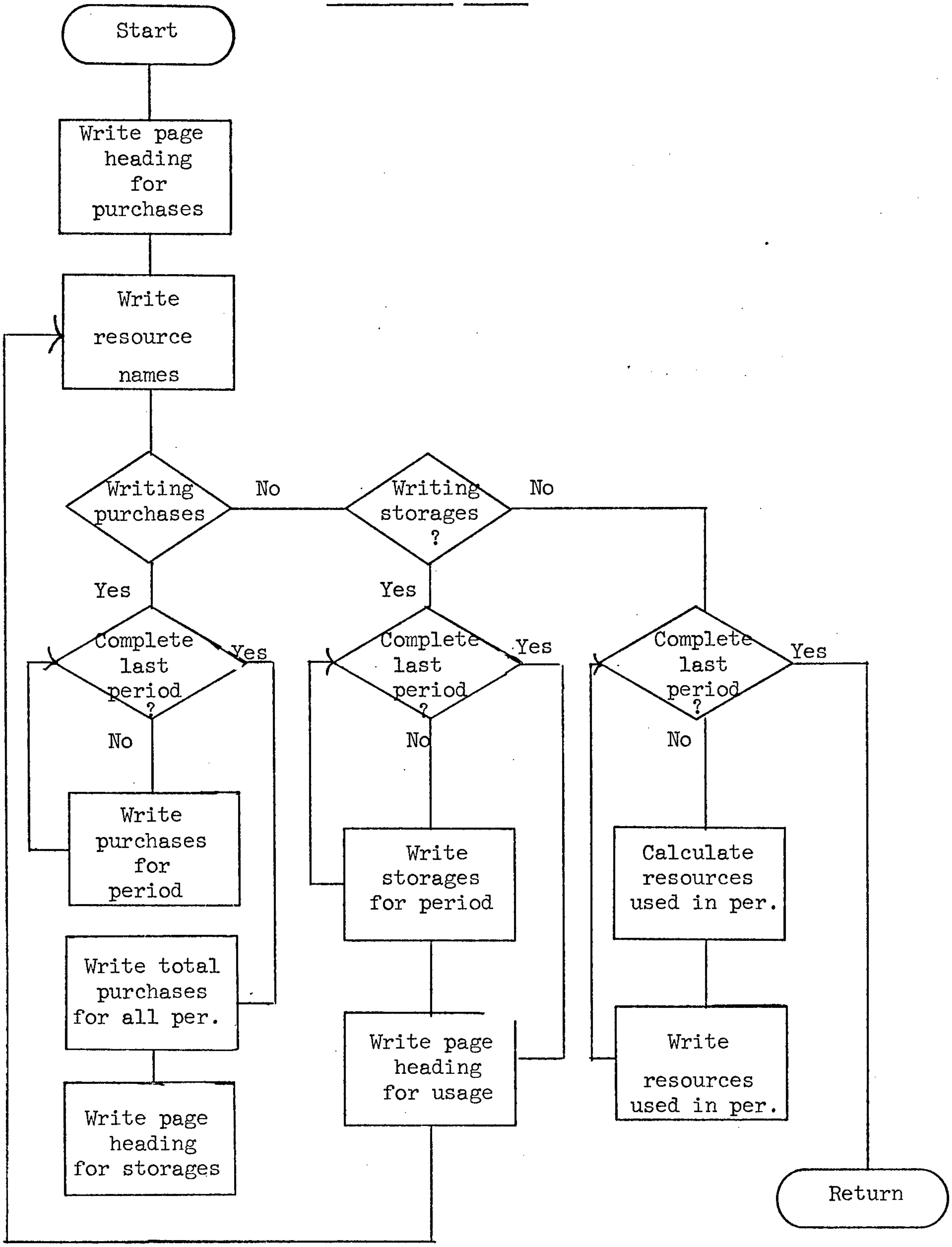

Start
Call
YRCOST*
Determine first & last year of existance
Increase O and M costs for those years
Increase resources for those years
Is last year the end of period?
Yes
No
Compute truncation values
Compute salvage values
Increase resources disposed of
Return

* This is the same routine as used in the matrix generator; its documentation is found there.

SUBROUTINE CINFO


Start
Write page heading for costs
Calculate actual & estimated procurement
Total all operating and salvage costs for per.
Correct proc. estimates, subtract salv. from operating
Sum all period costs, write costs for that per.
Add period costs to total costs
Last period ?
Yes
No
Add A & D to total costs
Write total costs
Return

SUBROUTINE PINFO


Start
Write page heading for purchases
Write resource names
Writing purchases
No
Writing storages ?
No
Yes
Yes
Complete last period ?
Yes
Complete last period ?
Yes
Complete last period ?
Yes
No
No
No
Write purchases for period
Write storages for period
Calculate resources used in per.
Write total purchases for all per.
Write page heading for usage
Write resources used in per.
Write page heading for storages
Return

APPENDIX D

PROGRAM LISTING

GENLCP & SUBROUTINES D-2

BBCAV2 & SUBROUTINES D-22

REPGEN & SUBROUTINES D-86

```
      PROGRAM GENLCP(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE4,TAPE9,
     *TAPE7)
C THIS PROGRAM GENERATES THE MATRIX FILE FOR THE LEAST COST PHASE-IN
C PROBLEM.
C
C  THE DIMENSIONS HAVE BEEN SET TO HANDLE
C   MAXIMUM NUMBER OF VEHICLES             =7
C   MAXIMUM VEHICLE LIFE     (IN YEARS)    =25
C   MAXIMUM NUMBER OF YEARS PRIOR TO SY=16
C   MAXIMUM NUMBER OF TASK TABLES          =8
C   MAXIMUM NUMBER OF ALTERNATIVES         =288
      COMMON /VECSTG/ VNAME(10), C,LENP, VLIFE(10), INH(10,16),
     * VCOST(10,5), NAMEN(10), COSTS(30,3)
      COMMON / ALTSTG / ALTER(288,9),YAVL(10)
      INTEGER ALTER
      INTEGER FNAME, SY,LY,VNAME,YAVL,VLIFE,YEAR(21)
      DIMENSION BUDG(10)
      DIMENSION NVEHU(20),NL(10),NN(10)
      DIMENSION NAMES(10), AU(16), UB(10),YRINT(20)
      COMMON /TSKSTG/ U(7,288,9) ,NTSK( 9)
      COMMON /PRDSTG/ NPERYR(10,3), NPTASK(10, 9), PTASK(10,9)
      DIMENSION IHVN(10)
      DIMENSION NP(288),NM(9),NPM(10)
C
      DATA(NP(I),I=1,240)/2H01,2H02,2H03,2H04,2H05,2H06,2H07,2H08,2H09,
     *2H10,2H11,2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,
     *2H23,2H24,2H25,2H26,2H27,2H28,2H29,2H30,2H31,2H32,2H33,2H34,2H35,
     * 2H36,2H37,2H38,2H39,2H40,2H41,2H42,2H43,2H44,2H45,2H46,2H47,2H48,
     * 2H49,2H50,2H51,2H52,2H53,2H54,2H55,2H56,2H57,2H58,2H59,2H60,
     * 2H61,2H62,2H63,2H64,2H65,2H66,2H67,2H68,2H69,2H70,2H71,2H72,
     * 2H73,2H74,2H75,2H76,2H77,2H78,2H79,2H80,2H81,2H82,2H83,2H84,
     * 2H85,2H86,2H87,2H88,2H89,2H90,2H91,2H92,2H93,2H94,2H95,2H96,
     * 2H97,2H98,2H99,2HA0,2HA1,2HA2,2HA3,2HA4,2HA5,2HA6,2HA7,2HA8,
     * 2HA9,2HB0,2HB1,2HB2,2HB3,2HB4,2HB5,2HB6,2HB7,2HB8,2HB9,2HC0,
     * 2HC1,2HC2,2HC3,2HC4,2HC5,2HC6,2HC7,2HC8,2HC9,2HD0,2HD1,2HD2,
     * 2HD3,2HD4,2HD5,2HD6,2HD7,2HD8,2HD9,2HE0,2HE1,2HE2,2HE3,2HE4,
     * 2HE5,2HE6,2HE7,2HE8,2HE9,2HF0,2HF1,2HF2,2HF3,2HF4,2HE5,2HF6,
     * 2HF7,2HF8,2HF9,2HG0,2HG1,2HG2,2HG3,2HG4,2HG5,2HG6,2HG7,2HG8,
     * 2HG9,2HH0,2HH1,2HH2,2HH3,2HH4,2HH5,2HH6,2HH7,2HH8,2HH9,2HJ0,
     * 2HJ1,2HJ2,2HJ3,2HJ4,2HJ5,2HJ6,2HJ7,2HJ8,2HJ9,2HK0,2HK1,2HK2,
     * 2HK3,2HK4,2HK5,2HK6,2HK7,2HK8,2HK9,2HL0,2HL1,2HL2,2HL3,2HL4,
     * 2HL5,2HL6,2HL7,2HL8,2HL9,2HM0,24M1,2HM2,2HM3,2HM4,2HM5,2HM6,
     * 2HM7,2HM8,2HM9,2HN0,2HN1,2HN2,2HN3,2HN4,2HN5,2HN6,2HN7,2HN8,
     * 2HN9,2HP0,2HP1,2HP2,2HP3,2HP4,2HP5,2HP6,2HP7,2HP8,2HP9,2HQ0/
      DATA(NP(I),I=241,288)/2HQ1,2HQ2,2HQ3,2HQ4,2HQ5,2HQ6,2HQ7,2HQ8,
     *2HQ9,2HR0,2HP1,2HP2,2HP3,2HP4,2HP5,2HR6,2HP7,2HP8,2HR9,2HT0,2HT1,
     *2HT2,2HT3,2HT4,2HT5,2HT6,2HT7,2HT8,2HT9,2HU0,2HU1,2HU2,2HU3,2HU4,
     *2HU5,2HU6,2HU7,2HU8,2HU9,2HW0,2HW1,2HW2,2HW3,2HW4,2HW5,2HW6,
     *2HW7,2HW8/
      DATA NM/2HM1,2HM2,2HM3,2HM4,2HM5,2HM6,2HM7,2HM8,2HM9/
      DATA NZ/2H00/
      DATA SX,SW,SP,SS,SB,SG/1HX,1HW,1HP,1HS,1HB,1HG/
      DATA IVT,ITT,IPT,IED /8HVEHICLE  ,8HTASK    ,8HPERIOD  ,
     * 8HENDTABLE /
      ONE=1.0
      ONEM=-1.0
C
```

```
      DO 3 I=1,7
      DO 2 J=1,288
      DO 1 K=1,9
      U(I,J,K)=0.0
    1 CONTINUE
    2 CONTINUE
    3 CONTINUE
C THE FIRST DATA CARD CONTAINS 1A. THE FILENAME TO BE USED =FNAME
C                              1B. THE STARTING YEAR (OR DECISION YEAR)
C                              1C. THE LAST YEAR = LY             =SY
C                              2A. THE NUMBER OF VEHICLES = NV
C                              2B. THE NUMBER OF TASKS    = NT
C                              2C. THE NUMBER OF PERIODS  = NPP
C                              3A. -P(L-1) PARAMETER
      READ(5,1000) FNAME, SY,LY, NV,NT,NPP,IZPLM1
 1000 FORMAT(A8,2X,6I5)
      WRITE(6,1010)FNAME, SY,LY, NV,NT,NPP
 1010 FORMAT(50H1 GENERATING THE MATRIX FOR THE LEAST COST PHASE-IN PROB
     *LEM   /11H0FILENAME= ,A8,16H STARTING YEAR =,I5,12H LAST YEAR =,I5,
     */11H WILL INPUT,I3,20H VEHICLE TABLES, AND, I3,16H TASK TABLE, AND
     *, I3,15H PERIOD TABLES. )
      NVR=0
      NTR=0
      NPT=0
      NIV=0
      NINHP=0
C  READ TITLE OF NEXT TABLE
   10 READ(5,1000)  ITABLE
C  DECIDE THE TYPE OF TABLE AND GO READ ITS DATA
      IF(ITABLE .EQ. IVT) GO TO 20
   11 IF(ITABLE .EQ. ITT) GO TO 40
      IF(ITABLE .EQ. IPT) GO TO 60
      IF(ITABLE .EQ. IEP) GO TO 100
C  THE TABLE NAME IS NOT RECOGNIZED, THERE IS AN INPUT ERROR
      WRITE(6,1020) ITABLE
 1020 FORMAT(1X,A8,60H IS NOT A TABLE NAME, INPUT ERROR. EXECUTION IS TE
     *RMINATED.     )
      STOP 1
   20 WRITE(6,1030)
 1030 FORMAT( 30H0 READING IN A VEHICLE TABLE     )
      NVR=NVR+1
      READ(5,1040)  VNAME(NVR), YAVL(NVR), VLIFE(NVR)
      WRITE(6,1050)VNAME(NVR), YAVL(NVR), VLIFE(NVR)
 1040 FORMAT(A8,1X,I4,6X,I2)
 1050 FORMAT(1X,A8,2X,2I10)
C YNAME=NAME OF VEHICLE, YAVL= 1ST YEAR VEHICLE AVALIBLE,
C YLIFE= MAXIMUM LIFE OF VEHICLE IN YEARS.
C
C IF THIS VEHICLE WAS AVALIABLE BEFORE THE STARTING YEAR, THEN READ IN
C SIZE OF INHERITED FLEET, BY YEAR BUILT.
      NU=SY - YAVL(NVR)
      IF( NU .LE. 0) GO TO 25
      NIV=NIV+1
      IA=1
      IB=8
   23 READ(5,1060)(INH(NVR,I),I=IA,IB)
 1060 FORMAT(8I10)
      IF(IB .GE. NU) GO TO 25
      IA=IB+1
      IB=IB+8
      GO TO 23
   25 READ (5,1070) (VCOST(NVR,I),I=1,5)
 1070 FORMAT (5F10.2)
C  VCOST(NVR,1) .GT. 1.0E30 INDICATES THIS VEHICLE IS NOT AVALIABLE FOR
C  PURCHASE.
      GO TO 10
C
C  IT IS ASSUMED THAT ALL VEHICLE TABLES ARE INPUTED FIRST.
   40 WRITE(6,1080)
 1080 FORMAT(25H0 READING IN A TASK TABLE  )
      NTR=NTR+1
      READ(5,1090)  IDT,NU,NA
      WRITE(6,1090) IDT,NU,NA
 1090 FORMAT(3I10)
C  IDT=TASK IDENTIFICATION NUMBER, NU=NUMBER OF VEHICLES,
C  NA=NUMBER OF ALTERNATIVES
      NTSK(IDT)=NA
      IA=1
      IB=8
   43 READ(5,1100) (NAMES(I),I=IA,IB)
 1100 FORMAT(8(A8,2X))
      IF(IB .GE. NU) GO TO 45
      IA=IB+1
      IB=IB+8
      GO TO 43
   45 DO 47 I=1,NU
      DO 46 J=1,NVR
      IF(VNAME(J) .NE. NAMES(I)) GO TO 46
      NAMES(I)=J
      GO TO 47
   46 CONTINUE
      WRITE(6,1110) NAMES(I)
 1110 FORMAT(15H0 VEHICLE NAME ,A8,60H NOT DEFINED IN A VEHICLE TABLE, E
     *XECUTION TERMINATED.          )
      STOP 2
   47 CONTINUE
C
C  NOW READ IN U(J,K,L), NUMBER OF VEHICLES OF TYPE J REQUIRED TO
C  PERFORM TASK L WITH ALTERNATIVE K.
      DO 55 K=1,NA
      IA=1
      IB=8
   48 READ(5,1120) (AU(I),I=IA,IB)
 1120 FORMAT(8F10.0)
      IF(IB .GE. NU) GO TO 49
      IA=IB+1
      IB=IB+8
      GO TO 48
   49 DO 50 I=1,NU
      J=NAMES(I)
      U(J,K,IDT)=AU(I)
   50 CONTINUE
   55 CONTINUE
   56 READ(5,1000) ITABLE
      GO TO 11
C
```

```
   60 WRITE(6,1130)
 1130 FORMAT(30H0 READING IN A PERIOD TABLE      )
C  THE PERIOD TABLES ARE EXPECTED IN CHRONOLOGICAL ORDER.
      NPT=NPT+1
      READ (5,1140) (NPERYR(NPT,I),I=1,2),BUDG(NPT)
 1140 FORMAT (I4,I5,3X,F8.2)
      WRITE(6,1150)(NPERYR(NPT,I),I=1,2)
 1150 FORMAT(2I5)
      IF(NPT .EQ. 1) GO TO 61
      IF(NPERYR(NPT-1,2)+1 .EQ. NPERYR(NPT,1) ) GO TO 61
      WRITE(6,1155)
 1155 FORMAT(35H THE PERIOD TABLES ARE OUT OF ORDER )
      STOP 3
   61 IF(SY .GT. NPERYR(NPT,1) ) GO TO 70
      READ(5,1158) NU,YRINT(NPT)
 1158 FORMAT (I10,F10.3)
C  ALL THE TASKS ARE SCALED BY THE FACTOR YRINT(NPT) IN THE  PERIOD NPT
      NPERYR(NPT,3)=NU
      IA=1
      IB=8
      NA=NPT-NINHP
   63 READ(5,1160)((NPTASK(NA,I), PTASK(NA,I)),I=IA,IB)
 1160 FORMAT(8(I5,F5.0))
      IF(IB .GE. NU) GO TO 56
      IA=IB+1
      IB=IB+8
      GO TO 63
   70 NINHP=NINHP+1
      GO TO 56
C
C  ALL TABLE HAVE BEEN READ IN. NOW PROCESS THEM TO BE ABLE TO GENERATE
C  THE MATRIX.
C  FIRST CHECK TO DETERMINE IF THE EXPECTED NUMBER OF TABLE WERE INPUTED
  100 IF((NV .EQ. NVR) .AND. (NT .EQ. NTR) .AND. (NPR.EQ. NPT))GO TO 105
      WRITE(6,1170)
 1170 FORMAT(71H WARNING-THE NUMBER OF TABLE ACTUALLY INPUT WAS NOT THE
     *EXPECT NUMBER.    )
C
C  ORDER THE VECHILES SO THE ARE IN DESENDING ORDER OF R+D COST.
  105 NRD=0
      DO 107 I=1,NVR
      NAMEN(I)=I
      IF(VCOST(I,3) .LE. 0.0) GO TO 107
      NRD=NRD+1
  107 CONTINUE
      IF(NRD.EQ.0) GO TO 151
      NV=NVR-1
      DO 120 II=1,NV
      I=NAMEN(II)
      IMAX=II
      IP1=II+1
      CMAX=VCOST(I,3)
      DO 110 JJ=IP1,NVR
      J=NAMEN(JJ)
      IF(CMAX .GE. VCOST(J,3))GO TO 110
      IMAX=JJ
      CMAX=VCOST(J,3)
  110 CONTINUE
```

```
      J=NAMEN(IMAX)
      NAMEN(IMAX)=NAMEN(II)
      NAMEN(II)=J
      IF(VCOST(J,3) .GT. 0.0) GO TO 120
      GO TO 125
  120 CONTINUE
C
C NOW DETERMINE IF FOR ANY R+D VEHICLE ITS DEVELOPMENT IS NOT OPTIONAL.
C  IT IS ASSUMED ALL TASKS ARE PREFORMED DURING SOME PERIOD AND
C  THE TASKS HAVE BEEN NUMBERED SEQUENTIALLY.
  125 DO 140 I=1,NRD
      NAMES(I)=1
      J=NAMEN(I)
      DO 133 L=1,NTR
      NA=NTSK(L)
      DO 130 K=1,NA
      IF(U(J,K,L) .EQ. 0.0) GO TO 133
  130 CONTINUE
C  FOUND A TASK REQUIRING THAT VEHICLE J BE DEVELOPED
      NAMES(I)=2
      GO TO 140
  133 CONTINUE
  140 CONTINUE
C  NAMES(I)=2 IF THE I TH MOST EXPENSIVE R+D COSTING VEHICLE MUST BE
C  DEVELOPED, =1 OTHERWISE
C  IF A VEHICLE MUST BE DEVELOPED TREAT IT AS IF ITS R+D COST =0
      NA=0
      DO 145 I=1,NRD
      K=NAMES(I)
      GO TO (145,143),K
  143 L=NAMEN(I)
      IP1=I+1
      K=NRD-NA
      DO 144 II=IP1,K
  144 NAMEN(II-1)=NAMEN(II)
      NAMEN(K)=L
      NA=NA+1
  145 CONTINUE
      NRD=NRD-NA
C  LIST VECHILE NAMES AND CORRESPONDING VARABLE LABELS.
      WRITE(6,1180)
 1180 FORMAT(33H0    VEHICLE NAME     VARIABLE NAME  /  8X,
     *                   21HOPTIONAL R+D VEHICLES  )
      DO 150 II=1,NRD
      I=NAMEN(II)
      WRITE(6,1190) VNAME(I), NP(II)
 1190 FORMAT(6X, A8, 5X, 1HX,A2)
  150 CONTINUE
      IF(NVR .LE. NRD) GO TO  200
  151 WRITE(6,1200)
 1200 FORMAT(13X,14HOTHER VEHICLES )
      J=NRD+1
      DO 155 II=J,NVR
      I=NAMEN(II)
      WRITE(6,1190) VNAME(I), NP(II)
  155 CONTINUE
C
      NROW=0
      MCOL=0
C LIST ROW NAMES
  200 WRITE(6,1210) FNAME
 1210 FORMAT(2H *, 4HNAME,11X,A8/ 2H *,4HROWS)
      WRITE(4,1211) FNAME
 1211 FORMAT      ( 4HNAME,10X,A8/ 4HROWS)
C
C
C  NOW THE ROW LABELS FOR THE MASTER VARIABLES
      DO 220 I=1,NVP
      WRITE(6,1240) NP(I)
 1240 FORMAT(2H *, 8H E   SUMX,A2)
      WRITE(4,1241) NP(I)
      NROW=NROW+1
 1241 FORMAT      ( 8H E   SUMX,A2)
  220 CONTINUE
C
C           ROWS FOR PROCUREMENT CONSTRAINTS
C
      IA=NPT-NINHP
      DO 225 I=1,IA
      WRITE (4,1225) NP(I)
 1225 FORMAT (6H E   PC,A2)
      WRITE (6,1224) NP(I)
 1224 FORMAT (2H *,6H E   PC,A2)
      NROW=NROW+1
  225 CONTINUE
C
C  NOW THE ROWS ACCOUNTING FOR THE INHERITED FLEET.
      IF(NINHP .EQ. 0) GO TO 300
      IB=NINHP - 1
      IF(IB .EQ. 0) GO TO 240
      DO 230 I=1,IB
      J=NINHP - I
      NPM(I)=NM(J)
  230 CONTINUE
  240 NPM(NINHP) =NZ
      IF(NIV .EQ. 0) GO TO 300
      NA=NPD + 1
      JC=1
      DO 260 JJ=NA,NVP
      J=NAMEN(JJ)
      IF(YAVL(J) .GE. SY) GO TO 260
      IHVN(JC)=JJ
      JC=JC+1
      DO 250 I=1,NINHP
      IF(YAVL(J) .GT. NPERYP(I,2)) GO TO 250
      IA=MAX0(YAVL(J),NPERYP(I,1)) - YAVL(J) + 1
      IB=NPERYP(I,2) - YAVL(J) + 1
      DO 245 K=IA,IB
      IF(INH(J,K) .GT. 0) GO TO 249
  245 CONTINUE
      GO TO 250
  249 WRITE(6,1250) NP(JJ),NPM(I)
 1250 FORMAT(2H *, 6H E   IW,A2,1HP,A2)
      WRITE(4,1251) NP(JJ),NPM(I)
      NROW=NROW+1
 1251 FORMAT      ( 6H E   IW,A2,1HP,A2)
```

```
  250 CONTINUE
  260 CONTINUE
C
C  NOW PUT OUT THE LABELS FOR THE ROWS FOR EACH PERIOD, THE VEHICLE
C  BALANCE ROWS FIRST, THEN THE TASK ROWS.
  300 IA=NINHP + 1
      DO 350 I=IA,NPT
      IB=I - NINHP
      NU=NPERYR(I,3)
      DO 340 JJ=1,NVR
C  IF THE VEHICLE IS NOT YET AVALIABLE IT CAN NOT BE USED.
      J=NAMEN(JJ)
      IF(YAVL(J) .GT. NPERYR(I,2)) GO TO 340
C  MAKE SURE THE VEHICLE IS USED
      DO 320 K=1,NU
      KT=NPTASK(IB,K)
      NA=NTSK(KT)
      DO 310 K2=1,NA
      IF(U(J,K2,KT) .NE. 0.0) GO TO 330
  310 CONTINUE
  320 CONTINUE
      GO TO 340
  330 WRITE(6,1260) NP(JJ), NP(IB)
 1260 FORMAT(2H *,5H E  X,A2,1HP,A2)
      WRITE(4,1261) NP(JJ), NP(IB)
      NROW=NROW+1
 1261 FORMAT     (5H E  X,A2,1HP,A2)
  340 CONTINUE
      DO 345 K=1,NU
      KT=NPTASK(IB,K)
      WRITE(6,1270) NP(KT), NP(IB)
 1270 FORMAT(2H *, 5H E  T,A2,1HP,A2)
      WRITE(4,1271) NP(KT), NP(IB)
      NROW=NROW+1
 1271 FORMAT     ( 5H E  T,A2,1HP,A2)
  345 CONTINUE
  350 CONTINUE
C
C  COMPUTE UPPER BOUNDS
      DO 390 II=1,NVR
      UB(II)=0.0
      I2=NAMEN(II)
      IA=NINHP+1
      DO 380 I=IA,NPT
      NU=NPERYR(I,3)
      I1=I - NINHP
      IF (YAVL(I2).GT.NPERYR(I,2)) GO TO 380
      DO 375 J=1,NU
      JJ=NPTASK(I1,J)
      TF= PTASK(I1,J)
      NA=NTSK(JJ)
      UMAX=0.0
      DO 370 K=1,NA
      IF(UMAX .GT. U(I2,K,JJ) ) GO TO 370
      UMAX=U(I2,K,JJ)
  370 CONTINUE
      UB(II)=UB(II) - TF*UMAX*YRINT(I)
  375 CONTINUE
  380 CONTINUE
  390 CONTINUE
      WRITE(6,1220)
 1220 FORMAT(2H *,8H N   COST)
      WRITE(4,1221)
      NROW=NROW+1
 1221 FORMAT      (8H N   COST)
C
      WRITE(6,1280)
 1280 FORMAT(2H *, 7HCOLUMNS)
      WRITE(6,1290)
 1290 FORMAT(2H *,8X,*(PARTIAL LISTING)*)
      WRITE(4,1281)
 1281 FORMAT      ( 7HCOLUMNS)
C  NOW GENERATE THE MATRIX ELEMENTS.
C
C  THE XNN COLUMNS.
      DO 420 I=1,NVP
      II=NAMEN(I)
      WRITE(6,1300) NP(I),NP(I),ONEM
 1300 FORMAT(2H *,4X,1HX,A2,7X,                                 4HSUMX,A2,4X,F12.4)
      WRITE(4,1301) NP(I),NP(I),ONEM
      MCOL=MCOL+1
 1301 FORMAT      (4X,1HX,A2,7X,                                4HSUMX,A2,4X,F12.4)
  420 CONTINUE
C
C        THE PNN COLUMNS
C
      IA=NPT-NINHP
      DO 430 I=1,IA
      WRITE (4,1311) NP(I),NP(I),ONE
 1311 FORMAT (4X,1HP,A2,7X,2HPC,A2,6X,F12.4)
      WRITE (6,1310) NP(I),NP(I),ONE
 1310 FORMAT (2H *,4X, 1HP,A2,7X,2HPC,A2,6X,F12.4)
      MCOL=MCOL+1
      IF (I.EQ.IA) GO TO 430
       IF(IZPLM1.EQ.1) GO TO 430
      WRITE (4,1313) NP(I),NP(I+1),ONEM
      WRITE (6,1312) NP(I),NP(I+1),ONEM
 1312 FORMAT (2H *,4X,1HP,A2,7X,2HPC,A2,6X,F12.4)
 1313 FORMAT       (4X,1HP,A2,7X,2HPC,A2,6X,F12.4)
  430 CONTINUE
C
C  GENERATE THE WJJLLMM COLUMNS
C
  440 IF(NIV .EQ. 0) GO TO 480
      DO 470 II=1,NIV
      JJ=IHVN(II)
      J=NAMEN(JJ)
      CALL YRCOST(J)
      DO 460 I=1,NINHP
      MAXL=VLIFE(J)
      IF(YAVL(J) .GT. NPERYR(I,2) ) GO TO 460
C  IT IS ASSUMED ALL THE VECHILES INHERITED FROM A PERIOD WERE PURCHASED
C  IN THE FIRST YEAR OF THE PERIOD.
      IA=MAX0(YAVL(J),NPERYR(I,1))- YAVL(J) + 1
      IB=NPERYR(I,2) - YAVL(J) + 1
      DO 445 K=IA,IB
      IF(INH(J,K) .GT. 0) GO TO 448
  445 CONTINUE
      GO TO 460
  448 NAGE=SY-NPERYR(I,1)
      C= COSTS(NAGE,2)
      LIFER=MAXL-NAGE
      IF(C .EQ. 0.0) GO TO 449
      C=-C
      WRITE(6,1330) NP(JJ),NPM(I),NZ, C
 1330 FORMAT(2H *,4X,1HW,A2,A2,A2,3X,4HCOST,6X,F12.4)
      WRITE(4,1331) NP(JJ),NPM(I),NZ, C
 1331 FORMAT     (4X,1HW,A2,A2,A2,3X,4HCOST,6X,F12.4)
  449 WRITE(6,1340) NP(JJ),NPM(I),NZ, NP(JJ),NPM(I),ONE
 1340 FORMAT(2H *,4X,1HW,A2,A2,A2, 3X, 2HIW,A2,1HP,A2,3X,F12.4)
      WRITE(4,1341) NP(JJ),NPM(I),NZ, NP(JJ),NPM(I),ONE
      MCOL=MCOL+1
 1341 FORMAT     (4X,1HW,A2,A2,A2, 3X, 2HIW,A2,1HP,A2,3X,F12.4)
      IA=NINHP+1
      DO 455 K=IA,NPT
C  MAKE SURE THE VECHILE IS USED
      KY=K-NINHP
      NU=NPERYR(K,3)
      DO 451 KK=1,NU
      KT=NPTASK(KY,KK)
      NA=NTSK(KT)
      DO 450 K2=1,NA
      IF(U(J,K2,KT) .NE. 0.0) GO TO 4511
  450 CONTINUE
  451 CONTINUE
      GO TO 455
 4511 IF(SY+LIFER .LE. NPERYR(K,1)) GO TO 460
      IY=NPERYR(K,2)-NPERYR(I,1)+1
      IX=NPERYR(K,2) -SY + 1
      C=-COSTS(IY,2)
      IF(K .EQ. NPT) C=-COSTS(IY,3)
      DO 452 KK=1,IX
      KKK=KK+NAGE
      C= C + COSTS(KKK,1)/VCOST(J,4)**KK
  452 CONTINUE
      WRITE(6,1330) NP(JJ),NPM(I),NP(KY), C
      WRITE(4,1331) NP(JJ),NPM(I),NP(KY), C
      WRITE(6,1340) NP(JJ),NPM(I),NP(KY), NP(JJ),NPM(I), ONE
      WRITE(4,1341) NP(JJ),NPM(I),NP(KY), NP(JJ),NPM(I), ONE
      MCOL=MCOL+1
      C=1.0
      ALPHA=VCOST(J,4)
      LLL3=0
      DO 4521 L3=IA,K
      L4=L3 - NINHP
      C=-C
      WRITE(6,1350) NP(JJ),NPM(I),NP(KY), NP(JJ),NP(L4), C
 1350 FORMAT(2H *,4X,1HW,A2,A2,A2,3X, 1HX,A2,1HP,A2,4X,F12.4)
      WRITE(4,1351) NP(JJ),NPM(I),NP(KY), NP(JJ),NP(L4), C
 1351 FORMAT     (4X,1HW,A2,A2,A2,3X, 1HX,A2,1HP,A2,4X,F12.4)
      LLL3=LLL3 + (NPERYR(L3,2)-NPERYR(L3,1) ) + 1
      C=ALPHA**LLL3
 4521 CONTINUE
  455 CONTINUE
  460 CONTINUE
  470 CONTINUE
C
C  GENERATE THE P, X, AND C COLUMNS FOR EACH PERIOD
  480 IA=NINHP + 1
      DO 600 LL=IA,NPT
C  IF YRINT(LL) .EQ. 1.0 IT IS ASSUMED ALL THE VEHICLES USED ARE
C  AVALIABLE.  HENCE NO CHECK IS MADE.
      IF (YRINT(LL).EQ. 1.0 ) GO TO 481
      NYR=NPERYR(LL,2)
C  THE SUBROUTINE YINTERR SETS THE ARRAY ALTER TO INDICATE THE
C  ALTERNATIVES THAT ARE NOT AVALIABLE FOR USE IN PERIOD LL.
C  IF ALTER(K,J)=0 THEN ALTERNATIVE J OF TASK K IS NOT AVALIABLE FOR
C  USE.
      CALL YINTERR (NVR,NTR,NYR)
  481 L=LL-NINHP
      DO 490 J=1,NVR
C
C  GENERATE THE PIIKKLL COLUMNS
  490 NVEHU(J)=1
      NU=NPERYR(LL,3)
      DO 520 II=1,NU
      ID=NPTASK(L,II)
      NA=NTSK(ID)
      KA=0
      DO 510 KK=1,NA
      IF (YRINT(LL).EQ. 1.0) GO TO 491
      IF (ALTER(KK,ID).EQ. 0) GO TO 510
  491 KA=KA+1
      DO 500 JJ=1,NVR
      J=NAMEN(JJ)
      IF( U(J,KK,ID)  .EQ. 0.0) GO TO 500
      IF (YAVL(J).GT.NPERYR(LL,2)) GO TO 500
      NVEHU(JJ)=2
      C=PTASK(L,II)*U(J,KK,ID)*YRINT(LL)
      WRITE(4,1361) NP(ID),NP(KA),NP(L), NP(JJ),NP(L), C
 1361 FORMAT     (4X,1HP,3A2,3X, 1HX,A2,1HP,A2, 4X, F12.4)
      IF (LL.NE.5) GO TO 500
      IF (KA.GT.10) GO TO 500
      WRITE(6,1360) NP(ID),NP(KA),NP(L), NP(JJ),NP(L), C
 1360 FORMAT(2H *,4X,1HP,3A2,3X, 1HX,A2,1HP,A2, 4X, F12.4)
  500 CONTINUE
      WRITE(4,1371) NP(ID),NP(KA),NP(L), NP(ID),NP(L), ONE
      MCOL=MCOL+1
 1371 FORMAT     (4X,1HP,3A2, 3X, 1HT,A2,1HP,A2, 4X, F12.4)
      IF (LL.NE.5) GO TO 510
      IF (KA.GT.10) GO TO 510
      WRITE(6,1370) NP(ID),NP(KA),NP(L), NP(ID),NP(L), ONE
 1370 FORMAT(2H *,4X,1HP,3A2, 3X, 1HT,A2,1HP,A2, 4X, F12.4)
  510 CONTINUE
  520 CONTINUE
      LENP=NPERYR(LL,2)-NPERYR(LL,1)+1
C
C  NOW GENERATE THE XJJLLMM COLUMNS
      DO 570 JJ=1,NVR
      IS=NVEHU(JJ)
      GO TO(570,525),IS
C  IS=2 INDICATES VEHICLE JJ IS USED IN PERIOD L
```

```
  525 J=NAMEN(JJ)
      CALL YRCOST(J)
      C=0
      DO 526 IS=1,LENP
  526 C=C + COSTS(IS,1)
      IF(NPERYR(LL,2).EQ. LY) GO TO 528
      C=C - COSTS(LENP,2)
      GO TO 527
  528 C=C - COSTS(LENP,3)
  527 WRITE(4,1391)  NP(JJ),NP(L),NP(L), NP(JJ), ONE
 1391 FORMAT      (4X,1HX,3A2, 3X, 4HSUMX,A2, 4X, F12.4)
      IF (LL.NE.5) GO TO 530
      WRITE(6,1390)  NP(JJ),NP(L),NP(L), NP(JJ), ONE
 1390 FORMAT(2H *,4X,1HX,3A2, 3X, 4HSUMX,A2, 4X, F12.4)
  530 WRITE(4,1401) NP(JJ),NP(L), NP(L), NP(JJ),NP(L), ONEM
 1401 FORMAT      (4X,1HX,3A2, 3X, 1HX,A2,1HP,A2,4X, F12.4)
      IF (LL.NE.5) GO TO 531
      WRITE(6,1400) NP(JJ),NP(L), NP(L), NP(JJ),NP(L), ONEM
 1400 FORMAT(2H *,4X,1HX,3A2, 3X, 1HX,A2,1HP,A2,4X, F12.4)
  531 IF (LL.NE.5) GO TO 529
      WRITE (6,1384) NP(JJ),NP(L),NP(L),NP(L),VCOST( J,5)
 1384 FORMAT (2H *,4X,1HX,3A2,3X,2HPC,A2,6X,F12.4)
      WRITE(6,1380) NP(JJ),NP(L),NP(L), C
 1380 FORMAT(2H *,4X,1HX,3A2, 3X, 4HCOST,6X,F12.4)
  529 WRITE (4,1385) NP(JJ),NP(L),NP(L),NP(L),VCOST( J,5)
 1385 FORMAT       (4X,1HX,3A2,3X,2HPC,A2,6X,F12.4)
      WRITE (4,1381) NP(JJ),NP(L),NP(L),C
      MCOL=MCOL+1
 1381 FORMAT      (4X,1HX,3A2, 3X, 4HCOST,6X,F12.4)
      LP1=LL + 1
      IF (LP1 .GT. NPT) GO TO 570
      DO 545 L1=LP1,NPT
C
C  MAKE SURE VEHICLE JJ IS USED IN PERIOD L1.
      IF( VLIFE(J) .LE. (NPERYR(L1,1) - NPERYR(LL,1)) ) GO TO 545
      NU=NPERYR(L1,3)
      DO 540 II=1,NU
      L2=L1-NINHP
      ID=NPTASK(L,II)
      NA=NTSK(ID)
      DO 535 KK=1,NA
      IF( U(J,KK,ID) .NE. 0.0) GO TO 5411
  535 CONTINUE
  540 CONTINUE
      GO TO 545
 5411 ALPHA=VCOST(J,4)
      C=1.0
      LLL3=0
      DO 5442 L3=LL,L1
      C=-C
      L4=L3-NINHP
      IF (LL.NE.5) GO TO 5443
      WRITE(6,1400) NP(JJ),NP(L),NP(L2), NP(JJ),NP(L4), C
 5443 WRITE(4,1401) NP(JJ),NP(L),NP(L2), NP(JJ),NP(L4), C
      LLL3=LLL3+ (NPERYR(L3,2)-NPERYR(L3,1)) + 1
      C=ALPHA**LLL3
 5442 CONTINUE
      C=0
      LLL1=NPERYR(L1,2)-NPERYR(LL,1) + 1
      DO 542 IS=1,LLL1
  542 C=C + COSTS(IS,1)
      IF(NPERYR(L1,2) .EQ. LY) GO TO 543
      C=C - COSTS(LLL1,2)
      GO TO 544
  543 C=C - COSTS(LLL1,3)
  544 WRITE (4,1385) NP(JJ),NP(L),NP(L2),NP(L),VCOST( J,5)
      WRITE (4,1381) NP(JJ),NP(L),NP(L2),C
      MCOL=MCOL+1
      IF (LL.NE.5) GO TO 5441
      WRITE (6,1384) NP(JJ),NP(L),NP(L2),NP(L),VCOST( J,5)
      WRITE(6,1380) NP(JJ),NP(L),NP(L2), C
 5441 WRITE(4,1391) NP(JJ),NP(L),NP(L2), NP(JJ), ONE
      IF (LL.NE.5) GO TO 545
      WRITE(6,1390) NP(JJ),NP(L),NP(L2), NP(JJ), ONE
  545 CONTINUE
C
C  NOW GENERATE THE SJJLL COLUMN
C
      CALL MOTH(J)
      WRITE(4,1411) NP(JJ),NP(L), C
 1411 FORMAT      (4X, 1HS,2A2,5X, 4HCOST,6X, F12.4)
      MCOL=MCOL+1
      WRITE (4,1412) NP(JJ),NP(L),NP(JJ),NP(L),ONE
 1412 FORMAT (4X,1HS,2A2,5X,1HX,A2,1HP,A2,4X,F12.4)
      IF (LL.NE.5) GO TO 570
      WRITE(6,1410) NP(JJ),NP(L), C, NP(JJ),NP(L), ONE
 1410 FORMAT(2H *,4X, 1HS,2A2,5X, 4HCOST,6X, F12.4, 3X,1HX,A2,1HP,A2,
     * 4X,F12.4)
  570 CONTINUE
  600 CONTINUE
C
C  NOW GENERATE THE RIGHT-HAND-SIDE ELEMENTS
      WRITE(6,1420)
 1420 FORMAT(2H *,3HRHS)
      WRITE(4,1421)
      MCOL=MCOL+1
 1421 FORMAT      (3HRHS)
C
C
C     GENERATE THE RHS FOR PROCUREMENT CONSTRAINTS
C
      IA=NPT-NINHP
      DO 610 I=1,IA
      IB=I+NINHP
      WRITE (4,1435) NP(I),BUDG(IB)
 1435 FORMAT       (4X,4HRHS1,6X,2HPC,A2,6X,F12.4)
      WRITE (6,1434) NP(I),BUDG(IB)
 1434 FORMAT (2H *,4X,4HRHS1,6X,2HPC,A2,6X,F12.4)
  610 CONTINUE
C
C  GENERATE THE RHS FOR INHERITED FLEET ROWS
  615 IF(NIV .EQ. 0) GO TO 650
      DO 640 II=1,NIV
      JJ=IHVN(II)
      J=NAMEN(JJ)
      DO 630 I=1,NINHP
      IF(YAVL(J) .GT. NPERYR(I,2)) GO TO 630
      ISUM=0
      IA=MAX0(YAVL(J),NPERYR(I,1)) - YAVL(J) + 1
      IB=NPERYR(I,2) - YAVL(J) + 1
      DO 620 K=IA,IB
  620 ISUM=ISUM + INH(J,K)
      IF(ISUM .EQ. 0) GO TO 630
      C=FLOAT(ISUM)
      WRITE(6,1440) NP(JJ), NPM(I), C
 1440 FORMAT(2H *,4X, 4HRHS1,6X, 2HIW,A2,1HP,A2,3X, F12.4)
      WRITE(4,1441) NP(JJ), NPM(I), C
 1441 FORMAT     (4X, 4HRHS1,6X, 2HIW,A2,1HP,A2,3X, F12.4)
  630 CONTINUE
  640 CONTINUE
C
C NOW GENERATE THE RHS FOR THE TASK ROWS
  650 IA=NINHP+1
      DO 700 LL=IA,NPT
      L=LL-NINHP
      NU=NPERYR(LL,3)
      DO 690 K=1,NU
      KT=NPTASK(L,K)
      WRITE(6,1450) NP(KT), NP(L), ONE
 1450 FORMAT(2H *,4X, 4HRHS1,6X, 1HT,A2,1HP,A2,4X,F12.4)
      WRITE(4,1451) NP(KT), NP(L), ONE
 1451 FORMAT     (4X, 4HRHS1,6X, 1HT,A2,1HP,A2,4X,F12.4)
  690 CONTINUE
  700 CONTINUE
      WRITE(6,1460)
 1460 FORMAT(2H *,  6HENDATA)
      WRITE(4,1461)
 1461 FORMAT     (  6HENDATA)
      END FILE 4
       CALL MATFILL(NROW,MCOL,UB,NVR)
      WRITE (6,3000) NROW,MCOL,(UB(I),I=1,NVR)
 3000 FORMAT (*0 IMPORTANT DATA ITEMS FOR INPUT TO BBCAVLP * /
     A * NUMBER OF ROWS (INCLUDING COST) IS *,I4 /
     B * NUMBER OF COLUMNS (INCLUDING RHS) IS *,I7 /
     C * UPPER BOUNDS FOR VEHICLES IN ORDER FROM X1 THRU XN ARE */
     D (1H ,10X,F12.4))
C
C   PRODUCE OUTPUT LISTING FOR DOCUMENTATION OF RUN
C
        WRITE (6,2010)
        WRITE (6,2020)
 2010 FORMAT (*1     VEHICLE   VARIABLE  PURCHASE     O AND M      R AND D
     *     RETENTION   YEAR FIRST  LIFE IN*)
 2020 FORMAT (*      NAME        NAME        COST          COST         COST
     *     RATE         AVAILABLE    YEARS*)
       IY=SY
C
C   LIST VEHICLE VARIABLE NAME, AND COST DATA
C
       DO 800 I=1,NVR
       II=NAMEN(I)
       WRITE (6,2030) VNAME(II),NP(I),(VCOST(II,J),J=1,4),YAVL(II),
      *VLIFE(II)
 2030 FORMAT (1H0,4X,A8,7X,1HX,A2,4(F8.4,4X),2X,I4,8X,I2)
```

```
      IF (YAVL(II).LT.IY) IY=YAVL(II)
  800 CONTINUE
C
C     DESCRIBE THE INHERITED FLEET
C
      IF (IY.EQ.SY) GO TO 821
      WRITE (6,2040)
 2040 FORMAT (*-     COMPONENTS OF THE INHERITED FLEET*)
      IF ((SY-IY).GT.20) IY=SY-20
      DO 810 I=IY,SY
      II=I-IY+1
  810 YEAR(II)=I
      INHYRS=SY-IY
      WRITE (6,2050) (YEAR(I),I=1,INHYRS)
 2050 FORMAT (1H0,20X,20(I5))
      NA=NVR-NIV+1
      DO 820 I= 1,NVR
      J=NAMEN(I)
      IF(YAVL(J).GE.SY) GO TO 820
      KK=YAVL(J)-IY
      DO 815 K=1,INHYRS
      IF (KK.LT.K) GO TO 814
      YEAR(K)=0
      GO TO 815
  814 K1=K-KK
      YEAR(K)=INH(J,K1)
  815 CONTINUE
      WRITE (6,2060) NR(I),(YEAR(K),K=1,INHYRS)
 2060 FORMAT (15H     NUMBER OF X,A2,4X,20(I5))
  820 CONTINUE
C
C     FOR EACH PERIOD , LIST ALL OF THE APPLICABLE TASK MATRICES
C
  821 IA=NINHP+1
      DO 850 I=IA,NPT
      WRITE (6,2070) NPERYR(I,1),NPERYR(I,2)
 2070 FORMAT  (35H-     TASKS REQUIRED IN PERIOD FROM ,I4,9H THROUGH ,I4)
      M=NPERYR(I,3)
      DO 845 J=1,M
      IM=I-NINHP
      JJ=NPTASK(IM,J)
      WRITE (6,2080) NP(JJ), PTASK(IM,J),  YRINT(I)
 2080 FORMAT (1H0,6X,*TASK *,A2,*  -   PERFORMED BY *,F5.2,* FORCE ELEMEN
     *T(S), WITH SCALE FACTOR EQUAL *,F5.3)
      II=0
      IF (YRINT(I).NE. 1.0) GO TO 845
      WRITE (6,2090)
 2090 FORMAT (1H ,6X,1H*)
C
C     DETERMINE WHICH VEHICLES ARE USED IN EACH TASK , JJ.....
C
C           (I=PERIOD, K=VEHICLE, II=NUMBER OF VEHICLES USED,
C           KK=NUMBER OF ALTERNATIVES)
C
      KK=NTSK(JJ)
      DO 830 K=1,NVR
      N=NAMEN(K)
      DO 829 L=1,KK
      IF (U(N,L,JJ).EQ.0) GO TO 829
      II=II+1
      NL(IT)=NP(K)
      NN(IT)=N
      NAMES(II)=K
      GO TO 830
  829 CONTINUF
  830 CONTINUE
      WRITE (6,2100) (NL(K),K=1,II)
 2100 FORMAT (1H ,7X,11H* VARIABLE ,10(3X,1HX,A2))
      WRITF (6,2110)
 2110 FORMAT (1H ,8X,9H*********)
      WRITE (6,2120)
 2120 FORMAT (1H ,6X,11HALTERNATIVE)
C
C    FILL IN TASK MATRIX
C
      DO 844 L=1,KK
      DO 840 K=1,IT
      N=NN(K)
      GO TO (831,832,833,834,835,836,837,838,839),K
  831 WRITE (6,2131) L,U(N,L,JJ)
 2131 FORMAT (1H ,15X,T2,2X,F5.0)
      GO TO 840
  832 WRITE (6,2132) U(N,L,JJ)
 2132 FORMAT (1H+,25X,F5.0)
      GO TO 840
  833 WRITE (6,2133) U(N,L,JJ)
 2133 FORMAT (1H+,31X,F5.0)
      GO TO 840
  834 WRITE (6,2134) U(N,L,JJ)
 2134 FORMAT (1H+,37X,F5.0)
      GO TO 840
  835 WRITE (6,2135) U(N,L,JJ)
 2135 FORMAT (1H+,43X,F5.0)
      GO TO 840
  836 WRITE (6,2136) U(N,L,JJ)
 2136 FORMAT (1H+,49X,F5.0)
      GO TO 840
  837 WRITE (6,2137) U(N,L,JJ)
 2137 FORMAT (1H+,55X,F5.0)
      GO TO 840
  838 WRITE (6,2138) U(N,L,JJ)
 2138 FORMAT (1H+,61X,F5.0)
      GO TO 840
  839 WRITE (6,2139) U(N,L,JJ)
 2139 FORMAT (1H+,67X,F5.0)
  840 CONTINUE
  844 CONTINUE
  845 CONTINUE
  850 CONTINUE
      STOP
      END
```

```
       SUBROUTINE MATFILL(N,M,UB,NVR)
      DIMENSION RVAL(120),RNAME(120)
      DIMENSION IROWTP(100)
       DIMENSION UB(1)
      DATA IT,II / 1HT,1HI /
      DATA C / 7HCOLUMNS /,R / 3HRHS  /
      I=0
      J=0
      DO 400 K=1,100
  400 IROWTP(K)=0
      REWIND 4
       WRITE(9,7000) M,N,(UB(I),I=1,NVR)
 7000  FORMAT(2I6/(6F12.4))
      READ (4,4000) DUM1,DUM2
      IF (EOF,4) 120,1
    1 WRITE(9,4000) DUM1,DUM2
 4000 FORMAT (A4,10X,A8)
      READ (4,4100) DUM3
 4100 FORMAT (A4)
      DO 10 I=1,N
      READ (4,4200) RNAME(I)
 4200 FORMAT (4X,A7)
      ENCODE(1,9000,ITEMP) RNAME(I)
      IF(ITEMP.EQ.IT.OR.ITEMP.EQ.II) IROWTP(I)=4
 9000 FORMAT(A1)
      IF (EOF,4) 120,10
   10 CONTINUE
      READ (4,4300)  DUM4
 4300 FORMAT (A7)
      IF (DUM4.EQ.C)GO TO 20
      WRITE  (6,4400) DUM4
 4400 FORMAT(* INCORRECTLY READ FILE----COLUMNS READ AS *,A7)
      RETURN
   20 READ (4,4500) CNAME,RTEMP,VAL
 4500 FORMAT (4X,A7,3X,A7,3X,F12.4)
      WRITE (6,5000)
 5000 FORMAT (*1 REFERENCE LIST FOR COLUMN NUMBERS AND NAMES*)
      WRITE(6,6100) (IROWTP(K),K=1,N)
      WRITE(9,6000) (IROWTP(K),K=1,N)
 6000 FORMAT(I12)
 6100 FORMAT(1H ,100I1)
      L=1
      DO 100 J=1,M
      GO TO (21,22,23,24,25),L
   21 WRITE (6,5100) J,CNAME
 5100 FORMAT (1H ,4X,I5,4X,A7)
      GO TO 26
   22 WRITE (6,5200) J,CNAME
 5200 FORMAT (1H+,24X,I5,4X,A7)
      GO TO 26
   23 WRITE (6,5300) J,CNAME
 5300 FORMAT (1H+,44X,I5,4X,A7)
      GO TO 26
   24 WRITE (6,5400) J,CNAME
 5400 FORMAT (1H+,64X,I5,4X,A7)
      GO TO 26
   25 WRITE (6,5500) J,CNAME
```

```
5500 FORMAT (1H+,84X,I5,4X,A7)
  26 L=L+1
     IF(L.GT.5)L=1
     WRITE(7,5700) J,CNAME
5700 FORMAT(I5,4X,A7)
     DO 30 I=1,N
  30 RVAL(I)=0.0
  40 DO 50 I=1,N
     IF (RTEMP.NE.RNAME(I)) GO TO 50
     RVAL(I)=VAL
     GO TO 60
  50 CONTINUE
  60 IF (J.NE.(M-1)) GO TO 80
     IF (I.NE.N) GO TO 80
     READ (4,4600) DUM5
4600 FORMAT (A3)
     IF (EOF,4) 120,70
  70 IF (DUM5.EQ.R) GO TO 80
     WRITE (6,4700) CNAME
4700 FORMAT(* THE M-1 COLUMN WAS *,A7,* ,UNABLE TO FIND RHS MARK*)
     RETURN
  80 READ (4,4500) CTEMP,RTEMP,VAL
     IF (EOF,4) 120,90
  90 IF (CTEMP.EQ.CNAME) GO TO 40
     CNAME=CTEMP
     WRITE (9,4800) (RVAL(K),K=1,N)
4800 FORMAT (F12.4)
 100 CONTINUE
     END FILE 9
     END FILE 7
     RETURN
 120 WRITE (6,4900) J,I
4900 FORMAT(* REACHED EOF WHILE WRITING COLUMN *,I7,* AND ROW *,I4)
     RETURN
     END
```

```
      SUBROUTINE YRCOST(J)
C  A SUBROUTINE TO COMPUTE THE OPERATING, SALVAGE, AND TRUNCATION
C  COSTS YEAR BY YEAR.  ALSO THE YEARLY MOTHBALLING SAVING IS COMPUTED.
      COMMON /VECSTG/ VNAME(10), C,LENP, VLIFE(10), INH(10,16),
     * VCOST(10,5), NAMEN(10), COSTS(30,3)
      INTEGER VNAME,VLIFE
C  ASSUME THE OPERATING AND MAINTANCE COST INCREACES AT R*100 PER-CENT
C  A YEAR (NOT A COMPOUND RATE INCEASE)
      R=0.0
C
C  LET X= THE 1ST YEAR O. AND M. COST.  THEN
C                                X+(1+R)*X+(+2*R)*X+...+(1+9*R)*X=VCOST(J,2)
      X= VCOST(J,2)/(10.0 + 45.0*R)
C  ASSUME NO PERIOD IS LONGER THAN 6 YEARS.
      IB=VLIFE(J) +10
      DO 10 I=1,IB
      COSTS(I,1)=(1.0 + FLOAT(I-1)*R)*X*(VCOST(J,4)**(I-1))
   10 CONTINUE
C
C  ASSUME THE SALVAGE VALUE OF A VEHICLE AFTER I YEARS OF SERVICE IS
C  (ALPHA)**I *PURCHASE COST.
      ALPHA=0.5
      Y=VCOST(J,1)
      DO 20 I=1,IB
      Y= ALPHA*Y
      COSTS(I,2)=Y
   20 CONTINUE
C
C          ASSUME TRUNCATION AFTER IYEARS OF SERVICE IS
C                 (VLIFE-I)*(PURCHASE COST)/VLIFE
C
      Y=VCOST(J,1)/VLIFE(J)
      DO 30 I=1,IB
      IX=VLIFE(J)-I
      IF (IX.LT.0) IX=0
      COSTS(I,3)=IX*Y
   30 CONTINUE
      RETURN
      ENTRY MOTH
C  ASSSUME THE MOTHBALLING SAVING IS R1*100 PER CENT OF THE FIRST YEAR COST-X
      R1=0.90
C     C=0
C     DO 546 IL=1,LENP
C 546 C=C-0.1*R1*VCOST(J,2)*VCOST(J,4)**(IL-1)
C     C =-X * R1
      C=-VCOST(J,2)/(10.0 + 45.0*R) * R1
      RETURN
      END
      SUBROUTINE YINTERP (NVR,NTR,NYR)
      COMMON /TSKSTG/ U(7,288,9) ,NTSK( 9)
      COMMON / ALTSTG / ALTER(288,9),YAVL(10)
      INTEGER ALTER
      INTEGER JSUB(10),YAVL
      DO 20 I=1,NTR
      N=NTSK(I)
      DO 10 J=1,N
   10 ALTER(J,I)=1
   20 CONTINUE
      DO 30 I=1,NVR
      IF (YAVL(I).LE.NYR) GO TO 30
      IVR=I
      GO TO 40
   30 CONTINUE
      RETURN
   40 L=0
      DO 50 J=IVR,NVR
      IF (YAVL(J).LE.NYR) GO TO 50
      L=L+1
      JSUB(L)=J
   50 CONTINUE
C
C  THE SET OF VEHICLES WHICH WILL NOT EXIST IN YEAR NYR
C     HAS BEEN DEFINED ---- NOW WE WILL ORDER THE SET
C     IN THE REVERSE OF THE ORDER IN WHICH THEY WILL
C     BE DEVELOPED......
C
      DO 70 I=1,L
      N=I
      K=JSUB(I)
      DO 60 J=N,L
      M=JSUB(J)
      IF (YAVL(M).LE.YAVL(K)) GO TO 60
      JSUB(J)=K
      JSUB(I)=M
      K=M
   60 CONTINUE
   70 CONTINUE
C
C  FOR EACH TASK, WE WILL DEFINE THE SET OF ALTERNATIVES
C     WHERE THE ≠NON-EXISTENT≠ VEHICLES ARE DOING ONLY
C     THOSE TASKS WHICH ARE THEIR PRIMARY RESPONSIBILITY,
C     THAT IS, WHERE THE REQUIREMENT FOR THEM IS A MINIMUM.....
C
      DO 150 I=1,NTR
      N=NTSK(I)
      DO 140 JJ=1,L
      J=JSUB(JJ)
      VMIN=9999.
      DO 100 K=1,N
      IF (ALTER(K,I).EQ.0) GO TO 100
      IF (U(J,K,I).LT.VMIN) VMIN=U(J,K,I)
  100 CONTINUE
      DO 130 K=1,N
      IF (ALTER(K,I).EQ.0) GO TO 130
      IF (U(J,K,I).EQ.VMIN) GO TO 130
      ALTER(K,I)=0
  130 CONTINUE
  140 CONTINUE
  150 CONTINUE
      RETURN
      END
      PROGRAM BBCAV2(INPUT,OUTPUT,TAPEA,TAPE1,TAPE2,
     1 TAPE3,TAPE7,TAPE8,TAPE5=INPUT,TAPE6=OUTPUT,
     2 TAPE9=TAPEA)
C
C  LABELLED  COMMON
      COMMON / CV1  /  IP(12),RP(12),TMP(10)
      COMMON / CV2  /  T(100,10),BO(100),BLO(10),ULO(10),CO(10)
      COMMON / CV3  /  M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
      COMMON / CV4  /  IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST
      COMMON / CV5  /  SIGMA(100,4),TSIG ,LSTMAX
      COMMON / CV7  /  NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
      COMMON / CV8  /  NXBK,XK,NOBOL,EKBL(25)
      COMMON / CV9  /  PSIGL(25),NXBL(25),XNXBL(25),BLIST(25,131)
       COMMON/TMX/TMO,EXT,TITLE(4)
C
       INTEGER UB,CI,BV
       DIMENSION TSTO(130),LSTFRE(25)
       DIMENSION BLT(10),ULT(10),CT(10)
C
C
    7 READ(5,4448)(TITLE(I),I=1,4)
      IF(EOF,5)1,2
    1 END FILE 8
      STOP0003
    2 CALL PARAMS
      RP(12)=0.0
      IP(9)=25
      NFREE=0
       CT  =  4
       UB  =  3
       LB  =  2
       BV  =  1
       MNC  =(-1)* NCF
       MNX  =  (-1)* N
       EPSI  =  RP(1)
       NORA  =  IP(2)
      MPLUS=NORA+NCF
       NOPS  =  1
       NCF4  =  NCF  *  3  +  NORA
C
      CALL READIN
       CALL  BOX1
C
C  SOLVE 1ST LP-PROBLEM
C
   55  CONTINUE
       US = USP
       IF( UZ .LT. 0.0) US = USM
      IF (NOP.GE.IP(12)) GO TO 4444
      LSTMAX=MAX0(LSTMAX,NOBOL)
      PMIN=1.E20
      DO 3000 I=1,NOBOL
      IF(PSIGL(I).GE.PMIN)  GO TO 3000
      PMIN=PSIGL(I)
      NMIN=I
 3000 CONTINUE
      IF(PMIN.LT.US)  GO TO 3020
      WRITE(6,3010)
 3010 FORMAT(1H-,14HPROBLEM SOLVED)
      GO TO 4446
 3020 PSIGL(NMIN)=1.E20
      NFREE=NFREE+1
      LSTFRE(NFREE)=NMIN
      NXBK=NXBL(NMIN)
      EKO=EKBL(NMIN)
      XK=XNXBL(NMIN)
      DO 3030 J=1,NCF
      J1=NORA+J
      J2=NCF+J1
      J3=NCF+J2
      BLO(J)=BLIST(NMIN,J1)
      ULO(J)=BLIST(NMIN,J2)
 3030  CO(J)=BLIST(NMIN,J3)
      DO 3040 J=1,NORA
 3040  BO(J)=BLIST(NMIN,J)
      INDIC=1
       BBK = BLO(NXBK)
       UBK = ULO(NXBK)
C
C
       DO 10 I = 1,NCF
       TMP(I) = 0.0
       BLT(I) = 0.0
       ULT(I) = 0.0
       CT(I) = 0.0
   10  CONTINUE
       DO 30 I = 1,NCF4
C
C  REDEFINE SIGMA FOR 1ST-LP-BB FROM KO-DATA
C
   30  TSTO(I) = 0.0
      NMIN=NORA-1
      DO 552 I=1,NMIN
  552  SIGMA(I,BV) = BO(I)
       DO 553 I = 1, NCF
       SIGMA(I,LB) = BLO(I)
       SIGMA(I,UB) = ULO(I)
       SIGMA(I,CI)= CO(I)
  553  CONTINUE
       XKSTO = XK
C                                                  SET   X(K) = Y(K)
       XK = XK - BBK
C                                                  SET UPPER BOUND = Y(K)
       SIGMA(NXBK,UB)= XK
       TSIG =EKO
      SIGMA(NORA,BV)=-TSIG
      BO(NORA)=-TSIG
C
       BLT(NXBK)= SIGMA(NXBK,LB)
       ULT(NXBK)=XK + BLT(NXBK)
C                                                  SLOPE OF X(NXBK)  (0 TO XK)
C                                                  SHIFTED RIGHT BY BLT(NXBK)
      BLO(NXBK)=BLT(NXBK)
      ULO(NXBK)=XK
       CALL GETC (NXBK,BLT,ULT,CT)
```

```
        SIGMA(NXBK,CI) = CT(NXBK)
      CO(NXBK)=CT(NXBK)
        NOP = NOP + 1
C                                    SOLVE K PRIME LP PROBLEM
      DO 5555 IND=1,MPLUS
      X(IND)=0
 5555 IX(IND)=0
      CALL TABOUT (1)
      NCF1=NCF
      NF1=0
      CALL LP (NOPA,N,NCF1)
      CALL TABOUT (2)
       CALL TIMEC
       COST1 = COST
       IF(NF1 .NE. 1)GO TO 90
   57  CONTINUE
      DO 6665 J=1,NCF
      TMP(J)=0
 6665 XCON(J)=0
      DO 6666 IND=1,MPLUS
      IF (IX(IND).GT.NCF .OR. IX(IND).EQ.0) GO TO 6666
      ICOL=IX(IND)
      TMP(ICOL)=X(IND)
      X(IND)=X(IND)+BLO(ICOL)
      XCON(ICOL)=X(IND)
C                                    DEFINE X(K) FROM Y(K)
 6666 CONTINUE
      IND=0
      DO 6677 J=1,NCF
      IF (BLO(J).EQ. 0.0) GO TO 6677
      IF (XCON(J).GT.0.0) GO TO 6677
      XCON(J)=BLO(J)
      IX(MPLUS-IND)=J
       X(MPLUS-IND)=BLO(J)
      IND=IND+1
 6677 CONTINUE
      RP(12)=COST-TSIG
      DO 6667 J=1,NCF
 6667 RP(12)=RP(12)-TMP(J)*CO(J)
       NOPS = NOPS + 1
       MNX = (-N)
       CALL TIMEC
C                                    EVALUATE OBJECTIVE F(X)
      CALL GETPHI (MNC,XCON,TMP,PHIT)
       CALL TIMEC
C
       WRITE(6,573) PHIT
  573 FORMAT(1H0,11HPHI(XADJ) =,1PE18.7)
      IF (IP(11).EQ.1)
     *WRITE (6,575) (IX(I),X(I),I=1,MPLUS)
  575 FORMAT (1H0,5(7H   COL ,I4,2H =,F12.4))
C
C
       IF(PHIT .GE. UZ)GO TO 70
C  PHIT .LT. UZ FOR 1ST-PROBLEM
C
       UZ = PHIT
      DO 58 I=1,MPLUS
      IXZ(I)=IX(I)
   58  XZ(I)   = X(I)
      NEWXZ=1
       USP = (UZ/(1.0 + EPSI))
       USM = (UZ/(1.0 - EPSI))
       US = USP
       IF(UZ .LT. 0.0)US = USM
   70  CONTINUE
       IF(COST1 .GE. US)GO TO 90
       CALL NXBRN(XCON, SIGMA, NXB)
 1990 IF(NFREE.LE.0) GO TO 2000
      NOL=LSTFRE(NFREE)
      NFREE=NFREE-1
      GO TO 2010
 2000 NOBOL=NOBOL+1
      NOL=NOBOL
      IF(NOBOL.LE.IP(9)) GO TO 2010
      WRITE(6,2020)
 2020 FORMAT(1H-,*BLIST SIZE EXCEEDED*)
      GO TO 4446
 2010 BSIGL(NOL)=COST
      NXBL(NOL)=NXB
      EKBL(NOL)=TSIG
      XNXBL(NOL)=XCON(NXB)
      DO 2030 J=1,NCF
      J1=NORA+J
      J2=NCF+J1
      J3=NCF+J2
      BLIST(NOL,J1)=SIGMA(J,LB)
      BLIST(NOL,J2)=SIGMA(J,UB)
 2030 BLIST(NOL,J3)=SIGMA(J,CI)
      DO 2040 J=1,NORA
 2040 BLIST(NOL,J )=SIGMA(J,BV)
      IF(INDIC.EQ.2) GO TO 55
   90  CONTINUE
      INDIC=2
       DO 91 I = 1,NCF
       BLT(I) = 0.0
       ULT(I) = 0.0
       CT(I)  = 0.0
       TMP(I)  = 0.0
   91  CONTINUE
C
C  REDEFINE SIGMA FOR 2ND-LP-PB FROM KO-DATA
C
      DO  95 I=1,NMIN
       SIGMA(I,BV) = BO(I)
       SIGMA(I,BV) = SIGMA(I,BV)- ( T(I,NXBK)*XK)
      BO(I)=SIGMA(I,BV)
   95  CONTINUE
       DO 96 I = 1,NCF
       SIGMA(I,LB) = BLO(I)
       SIGMA(I,UB) = ULO(I)
       SIGMA(I,CI) = CO(I)
C                                        DEFINE LOWER BOUND OF X(K)
C                                        IF BBK = 0
   96  CONTINUE
C                                        SET UPPER BOUND OF Y(K).
```

```
C                                    THIS IS THE ONLY BOUND FOR
C                                    THIS VARIABLE SENT TO THE LP CODE
       BBK2 = BBK + XK
       UBK2 = UBK-XK
       SIGMA(NXBK,UB) = UBK2
       SIGMA(NXBK,LB) = BBK2
       BLT(NXBK) = BBK
       CALL GETPHI(NXBK,BLT,TMP,DMY)
       PH1 = TMP(NXBK)
       BLT(NXBK) = BBK2
       CALL GETPHI(NXBK,BLT,TMP,DMY)
       PH2 = TMP(NXBK)
       TP2 = 0
       TSIG = EKO  - PH1 + PH2
      SIGMA(NORA,BV)=-TSIG
      BO(NORA)=-TSIG
       BLT(NXBK) = BBK2
       ULT(NXBK) = BBK2 + UBK2
C                                    SET SLOPE OF X(K), IF BBK = 0
      BLO(NXBK)=BLT(NXBK)
      ULO(NXBK)=UBK2
       CALL GETC (NXBK,BLT,ULT,CT)
       SIGMA(NXBK,CI) = CT(NXBK)
      CO(NXBK)=CT(NXBK)
       NOP = NOP + 1
C                                    SOLVE K DOUBLE PRIME LP PROBLEM
      DO 7777 IND=1,MPLUS
      X(IND)=0
 7777 IX(IND)=0
      CALL TABOUT (1)
      NCF1=NCF
      NF1=0
      CALL LP (NORA,N,NCF1)
      CALL TABOUT (2)
       COST2 = COST
       IF(NF1 .NE. 1)GO TO 55
  104  CONTINUE
       NOPS = NOPS + 1
      DO 8887 J=1,NCF
      TMP(J)=0
 8887 XCON(J)=0
      DO 8888 IND=1,MPLUS
      IF (IX(IND).GT.NCF .OR. IX(IND).EQ.0) GO TO 8888
      ICOL=IX(IND)
      TMP(ICOL)=X(IND)
      X(IND)=X(IND)+BLO(ICOL)
      XCON(ICOL)=X(IND)
 8888 CONTINUE
      IND=0
      DO 8899 J=1,NCF
      IF (BLO(J).EQ. 0.0) GO TO 8899
      IF (XCON(J).GT.0.0) GO TO 8899
      XCON(J)=BLO(J)
      IX(MPLUS-IND)=J
       X(MPLUS-IND)=BLO(J)
      IND=IND+1
 8899 CONTINUE
      RP(12)=COST-TSIG
      DO 8889 J=1,NCF
 8889 PP(12)=PP(12)-TMP(J)*CO(J)
      CALL GETPHI (MNC,XCON,TMP,PHIT)
C
       WRITE(6,573) PHIT
      IF (IP(11).EQ.1)
     *WRITE (6,575) (IX(I),X(I),I=1,MPLUS)
C
       IF(PHIT .GE. UZ)GO TO 109
       UZ = PHIT
      DO 107 I=1,MPLUS
      IXZ(I)=IX(I)
  107  XZ(I)     =    X(I)
      NEWXZ=1
       USP = ( UZ /(1.0 + EPSI))
       USM = ( UZ /(1.0 - EPSI))
       US = USP
       IF(UZ .LT. 0.0)US = USM
  109  CONTINUE
      IF(COST2.GE.US) GO TO 55
       CALL MXRPN(XCON, SIGMA, NXP)
      GO TO 1990
 4444 WRITE (6,4445)
 4445 FORMAT (* HAVE SOLVED MAX. NO. OF LP PROBS. SET BY IP(12)*)
 4446 WRITE(8,4448)(TITLE(I),I=1,4)
 4448 FORMAT(4A10)
      WRITE(8,4447)(IXZ(I),XZ(I),I=1,MPLUS)
 4447 FORMAT(I4,4X,F12.4)
      WRITE(8,4447)MNC,UZ
      NEWXZ=1
      CALL TABOUT (3)
      GO TO 7
   26  CALL EXIT
       END
      SUBROUTINE BOX1
C
C  LABELLED  COMMON
      COMMON / CV1 / IP(12),RP(12),TMP(10)
      COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)
      COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
      COMMON / CV4 / IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST
      COMMON / CV5 / SIGMA(100,4),TSIG ,LSTMAX
      COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
      COMMON / CV8 / NXBK,XK,NOBOL,EKBL(25)
      COMMON / CV9 / PSIGL(25),NXBL(25),XNXBL(25),BLTST(25,131)
C
C
      INTEGER UB,CI,BV
C
C  BOX NO. 1   (NOP = 1)
C
       CI = 4
      UB = 3
      LB = 2
      BV = 1
      NORA = IP(2)
      MNC =(-1)* NCF
      MNX = (-1)* N
     CALL GETC (MNC,BLO,ULO,CO)
     CALL INITA (NCF,N,NORA)
      CALL GETPHI(MNC,BLO,TMP,ESIG)
      EKO = ESIG
C  SET ISIGMA FOR 1ST LP PROB.
      DO 10 I = 1,NCF
      TMP(I) = 0.0
       SIGMA(I,LB)=BLO(I)
      SIGMA(I,UB)=ULO(I)
      SIGMA(I,CI)=CO(I)
   10 CONTINUE
      DO 15 I = 1,NORA
      SIGMA(I,BV)= BO(I)
   15 CONTINUE
      TSIG = EKO
C
       NOP = 1
      DO 5555 IND=1,MPLUS
      XZ(IND)=0
      IXZ(IND)=0
      X(IND)=0
 5555 IX(IND)=0
      CALL TABOUT (1)
      NCF1=NCF
      NF1=0
      CALL LP (NORA,N,NCF1)
      CALL TABOUT (2)
       IF(NF1 .NE. 1)GO TO 7
   28  CONTINUE
      DO 31 J=1,NCF
   31 XCON(J)=0
      DO 6666 IND=1,MPLUS
      IF (IX(IND).GT.NCF .OR. IX(IND).EQ.0) GO TO 6666
      ICOL=IX(IND)
      X(IND)=X(IND)+BLO(ICOL)
      XCON(ICOL)=X(IND)
 6666 CONTINUE
      DO 30 J=1,MPLUS
      IXZ(J)=IX(J)
       XZ(J)  = X(J)
   30  CONTINUE
      NEWXZ=1
      RP(12)=COST
      DO 6667 J=1,NCF
 6667 RP(12)=RP(12)-XCON(J)*CO(J)
      CALL GETPHI (MNC,XCON,TMP,UZ)
       EPSI  = RP(1)
       USP=  (UZ /(1.0  + EPSI))
       USM=  (UZ /(1.0  - EPSI))
       EKO  =  TSIG
C
C   10  SEP  68
      CALL NXBRN (XCON,SIGMA,NXB)
      LSTMAX=1
      NOBOL=1
      PSIGL(1)=COST
      XK=XCON(NXB)
      XNXBL(1)=XCON(NXB)
      NXBL(1)=NXB
      EKBL(1)=TSIG
   50  CONTINUE
       DO 52  I  =  1,NCF
       BLO(I)  =  SIGMA(I,LB)
       CO(I)    =  SIGMA(I,CI)
       ULO(I)  =  SIGMA(I,UB)
      I1=NOPA+I
      I2=NCF+I1
      I3=NCF+I2
      BLIST(1,I1)=BLO(I)
      BLIST(1,I2)=ULO(I)
      BLIST(1,I3)=CO(I)
   52  CONTINUE
       DO 53  I  =  1,NOPA
       BO(I)    =  SIGMA(I,BV)
      BLIST(1,I )=BO(I)
   53  CONTINUE
  777  RETURN
    7 CALL BBCAV2
       END
      SUBROUTINE GETASQ(NOES,ELM,JSQ)
C...SHELL METHOD OF HALVING
C
C   GETASQ(NOES,ELM,JSQ)   SORTS ELM(J),J=1,NOES IN AN ASCENDING SEQUENCE
C   PRESET INITIAL POSITION CODE OF ELM(J)
C   JSQ(J) PRESET TO (-1) WHEN ELM(J) IS UNDEFINED (I.E. INFINITE)
C
       DIMENSION ELM(1), JSQ(1)
       L = 1
    7  L = 2 * L
       IF( L.LE. NOES) GO TO 7
       L = L - 1
   10  L = L / 2
       DO 20 K2 = 1, NOES
       K1 = K2
   15  K3 = K1 + L
       IF( K3 .GT. NOES) GO TO 30
       IF( ELM(K1).LE. ELM(K3) ) GO TO 20
       RT = ELM(K1)
       ELM(K1 ) = ELM(K3)
       ELM(K3) = RT
       RT = JSQ(K1)
       JSQ(K1) = JSQ(K3)
       JSQ(K3) = RT
       K1 = K1 - L
       IF( K1 .GE. 1) GO TO 15
   20  CONTINUE
   30  IF( L .GT. 1) GO TO 10
       RETURN
      END
      SUBROUTINE GETC (KCX,BLT,ULT,CT)
     COMMON / CV1 / IP(12),RP(12),TMP(10)
     COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
      DIMENSION BLT(01),ULT(01),CT(01),FX1(10),FX2(10)
C
C  IF  (KCX) 1. .GT. 0, EVALUATE KCX(TH) C(X)-SLOPE.
C            2. .LT. 0, EVALUATE CX(1)  TO CX(IFX), (IFX = -KFX).
C            3. .EQ. 0, INVALID KCX **** UEP.
C
      IF(IP(6) .EQ. 1) WRITE(6,999)
  999 FORMAT(1H-,124X,6HGETC  )
      IF(KCX .GT. N)  GO TO 770
      IF(KCX)200,770,100
  100 FX1(KCX) = 0.0
     FX2(KCX) = 0.0
      CALL GETPHI(KCX,BLT,FX1,DMY)
      CALL GETPHI(KCX,ULT,FX2,DMY)
      NDX1 = KCX
      NDX2 = KCX
      GO TO 220
  200 ICX =(-1) * KCX
      IF( ICX .GT. N) GO TO 770
     DO 210 I = 1,ICX
      FX1(I) = 0.0
      FX2(I) = 0.0
      CT(I)  = 0.0
  210 CONTINUE
      CALL GETPHI(KCX,BLT,FX1,DMY)
      CALL GETPHI(KCX,ULT,FX2,DMY)
      NDX1 = 1
      NDX2 = ICX
  220 DO 225 J = NDX1,NDX2
      DIF = ULT(J) - BLT(J)
      IF(DIF    .EQ. 0.0) GO TO 225
      CT(J) = (FX2(J) - FX1(J)) / DIF
  225 CONTINUE
      GO TO 777
  770 WRITE(6,771)KCX
  771 FORMAT(1H1,13HINVALID KCX =,I3,10H  IN GETC    )
      CALL EXIT
C
  777 CONTINUE
  888 RETURN
C
      END
      SUBROUTINE GETPHI(KFX,XPHI,PHI,SUMPHI)
      DIMENSION XPHI(01),PHI(01)
     COMMON / CV1 / IP(12),RP(12),TMP(10)
     COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)
     COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
C
C  IF (KFX)   1. .GT.  0, EVALUATE  KFX(TH) F(X).
C             2. .LT.  0, EVALUATE  FX(1) TO FX(IFX), (IFX = -KFX).
C             3. .EQ.  0, INVALID KFX **** UEP.
C
      IF(IP(6) .EQ. 1) WRITE(6,999)
  999 FORMAT(1H-,124X,6HGETPHI)
C
      IF(KFX)100,300,500
  100 SUMPHI=RP(12)
      I = 1
  160 IF(I+KFX) 150,150,400
  150 IF(I.GT.4)GO TO 140
      GO TO (101,102,103,104) I
  101 IF(XPHI(I).LT..0001) GO TO 140
      PHI(I)= 0.30 + 0.006*XPHI(I)**0.95
      GO TO 200
  102 PHI(I)= 0.0038*XPHI(I)**0.96
      GO TO 200
  103 PHI(I)= 0.006*XPHI(I)**0.90
      GO TO 200
  104 PHI(I)= 0.015*XPHI(I)**0.909
      GO TO 200
  140 PHI(I) = 0.0
  200 SUMPHI = SUMPHI + PHI(I)
      IF(KFX.GT.0) RETURN
      I = I+1
      GO TO 160
  500 SUMPHI = 0.0
      I = KFX
      GO TO 150
  300 WRITE(6,301)
  301 FORMAT(1H1,25HKFX = 0 IN GETPHI         )
      CALL EXIT
  400 RETURN
       END
      SUBROUTINE INITA(NCF,N,M)
C
C         THIS SUBROUTINE COPIES THE A MATRIX FROM TAPE TO DISC
C         AND STORES THE BO AND CO ARRAYS IN CORE. TAPE9 IS ASSUMED
C         TO BE THE TAPE AND TAPE3 IS THE DISC FILE.
C
      COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)
      COMMON /ROWTYP/ IROWTP(101)
      DIMENSION AJ(100)
      REWIND 3
C     REWIND 9
      READ (9,100) DUM1,DUM2
  100 FORMAT (A4,10X,A8)
      READ(9,400)(IROWTP(J),J=1,M)
  400 FORMAT(I12)
      DO 10 I=1,N
      READ (9,200)  (AJ(J),J=1,M)
  200 FORMAT (F12.4)
      IF(EOF,9) 1000,20
   20 IF (I.NE.N) GO TO 40
      DO 30 J=1,M
   30 BO(J)=AJ(J)
   40 IF (I.GT.NCF) GO TO 60
      AJ(M)=CO(I)
      DO 55 J=1,M
   55 T(J,I)=AJ(J)
   60 WRITE (3)      (AJ(J),J=1,M)
C      WRITE(7,1) (AJ(J),J=1,M)
1      FORMAT(5F15.5)
C      WRITE(6,2) (AJ(J),J=1,M)
2      FORMAT(1X,5E15.6)
   10 CONTINUE
      IROWTP(M)=3
      END FILE 3
      RETURN
 1000 WRITE (6,300) I
  300 FORMAT (* PREMATURE EOF ON A MATRIX TAPE AT COLUMN *,I5)
      STOP0002
      END
```

```
       SUBROUTINE NXBRN(XT,SIGMAJ,NXB)
      COMMON / CV1 / IP(12),RP(12),TMP(10)
      COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
      DIMENSION XT(001),BLT(10),CT(10),  YT(10),SIGMAT(100,4)
      DIMENSION  FX1(10),FX2(10),DIF(10),NDX(10)
   10  CONTINUE
       IF(IP(6) .EQ. 1) WRITE(6,999)
  999  FORMAT(1H-,124X,6HNXBRN )
       NXB =  0
C  NXBRN GIVES BEST BRNCH-CANDIDATE FOR  XT(I)
       DO 5  J = 1,NCF
       FX2(J)= 0.0
       FX1(J)= 0.0
       DIF(J)= 0.0
       NDX(J)= 0
       BLT(J)=  SIGMAT(J,2)
       CT(J)  =  SIGMAT(J,4)
    5  CONTINUE
       DO 20 J = 1,NCF
       YT(J)  = XT(J)  - BLT(J)
   20  CONTINUE
       NFX =(-1) * NCF
       CALL GETPHI(NFX,  XT,FX2,DMY)
       CALL GETPHI(NFX,BLT,FX1,DMY)
   40  CONTINUE
      IF (RP(4).NE.0)
     *WRITE (6,55)
   55 FORMAT(1H0,10X,*DIFFERENCE    =*,10X,*PHI(X)    -   PHI(LOWER BOUND)
     * -   (*,8X,4HC(X),4X,1H*,12X,1HX,6X,1H))
       DO 30 J = 1,NCF
       DIF(J) = FX2(J) - FX1(J) - CT(J)*YT(J)
      IF (RP(4).NE.0)
     * PRINT 50,J,DIF(J),FX2(J),FX1(J),CT(J),YT(J)
   50  FORMAT(1H0,I5,6F20.6)
       NDX(J) = J
   30  CONTINUE
       CALL GETASQ(NCF,DIF,NDX)
       NXB = NDX(NCF)
       RETURN
 1000  CONTINUE
       END
      SUBROUTINE PARAMS
C
C  LABELLED COMMON
      COMMON / CV1 / IP(12),RP(12),TMP(10)
      COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)
      COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
      COMMON / CV4 / TX(110),X(110),TXZ(110),XZ(110),XCON(10),COST
      COMMON / CV5 / SIGMA(100,4),TSIG,IFIL
      COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
      COMMON / CV8 / NXBK,XK,NOBOL,EKBL(25)
      COMMON / CV9 / PSIGL(25),NXBL(25),XNXBL(25),BLIST(25,131)
C
       READ(5,10)(IP(I),I=1,12)
   10  FORMAT(12I6)
       REWIND 9
       READ(9,11) IP(1), IP(2)
11     FORMAT(2I6)
       IF( EOF, 5) 77777, 15
   15  CONTINUE
C
C  IP1=N, IP2=NORA, IP3=NCF, IP4=MMAX, IP5=NMAX, IP6=LPIN, IP7=LPOUT
C  IP8=ICTU, IP9=IBMAX, IP10=JBMAX, IP11=ICK, IP12=MXNOP
C
C  N - TOTAL NO. OF VARIABLES
C  NORA - NO. OF ROWS IN A-MATRIX
C  NCF - NO. OF VARIABLES W/CONCAVE-F(X)
C  MMAX - MAX. NO. OF CONSTRAINTS FOR JPLP
C  NMAX - MAX. NO. OF VARIABLES FOR JPLP
C  LPIN - IF (1) WRITE LP INPUT FOR EACH PROBLEM
C  LPOUT - IF (1) WRITE LP OUTPUT FOR EACH PROBLEM
C  ICTU - IF (1) CONSTRAIN COST-F(X) .LT. U0
C  IBMAX - MAX NO. OF ROWS IN BLIST
C  JBMAX - MAX NO. OF COLUMNS IN BLIST
C  ICK - IF (1) SET PRINT = .TRUE. IN JPLP
C  MXNOP - MAX. NO. OF LP PROBS. SOLVED BEFORE CALLING EXIT
C
       READ(5,20)(RP(I),I=1, 6)
   20  FORMAT( 6E12.0)
C
C  RP1=EPSI, RP2=TMMAX, RP3=THETA, RP4=TRACE
C  EPSI - ADJUSTMENT FACTOR FOR U0
C  TMMAX - MAX.  BB-EXCT TIME IN SECONDS
C  THETA - X(I) ZERO RNDOFF
C  TRACE - IF(1) TRACE SOLUTION, USING LPIN, LPOUT, AND ICK CODES
C          IF(0) SKIP ALL INTERMEDIATE PRINT OUT
C
       TMMAX = RP(2)
       CALL SET(TMMAX)
C
       CALL PRESET
C
       WRITE(6,30)(IP(I),I=1,12)
   30  FORMAT(1H1,20HINTEGER PARAMETERS =,12I6)
       WRITE(6,40)(RP(I),I=1,6)
   40  FORMAT(1H-,17HREAL PARAMETERS =,6E18.8)
       N = IP(1)
       NORA = IP(2)
      NCF = IP(3)
      M=NCF + NORA
C
      IBMAX = IP(9)
C
   55 READ(9,20)(        ULO(J), J = 1,NCF)
      DO 60 J=1,NCF
      IF(ULO(J).LT.0) ULO(J) = - ULO(J)
60    CONTINUE
C     READ(5,20) (BLO(J),J=1,NCF)
      DO 61 J=1,NCF
C
61    BLO(J) = 0.0
      WRITE(6,90)
   90 FORMAT(1H-,25H X(J) LOWER-UPPER BOUNDS )
      DO 100 J = 1,NCF
      WRITE(6,95)J,BLO(J),ULO(J)
   95 FORMAT(1H0,2X,I3,3X,2E12.4)
  100 CONTINUE
      NOBOL = 0
  777 RETURN
77777 CONTINUE
      STOP 00001
      END
      SUBROUTINE PRESET
C
C LABELLED COMMON
     COMMON / CV1 / IP(12),RP(12),TMP(10)
     COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)
     COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
     COMMON / CV4 / IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST
     COMMON / CV5 / SIGMA(100,4),TSIG
     COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
     COMMON / CV8 / NXBK,XK,NOBOL,EKBL(25)
     COMMON / CV9 / PSIGL(25),NXBL(25),XNXBL(25),BLIST(25,131)
C
      IF(IP(6) .EQ. 1) WRITE(6,999)
  999 FORMAT(1H-,124X,6HPRESET)
      N=IP(1)
      NOPA = IP(2)
      NCF=IP(3)
      IBMAX = IP(9)
      JBMAX = IP(10)
C
C
      DO 13 J=1,NCF
      CO(J) = 0.0
      BLO(J) = 0.0
      ULO(J) = 0.0
   13 CONTINUE
C
      DO 15 I = 1,NOPA
      DO 14 K=1,4
   14 SIGMA(I,K)=0.0
      BO(I) = 0.0
   15 CONTINUE
C
     NEWXZ=0
      PHIT=0
      UZ = 1.E+35
      USP = 1.E+36
      USM = 1.E+36
      COST=0.0
      TSIG=0.0
      NF1=0
      CFX=0.0
      IOPT=0
      NOP=0
     NXBK=0
      XK =0.0
      NOBOL=0
C
      DO 20 I =1,IBMAX
      EKBL(I) = 0.0
      PSIGL(I)= 0.0
      NXBL(I)= 0
      XNXBL(I)=0.0
      DO 20 J=1,JBMAX
      BLIST(I,J) = 0.0
   20 CONTINUE
C
```

```
RETURN                                                      20007640
END                                                         20007650
```

```
      SUBROUTINE READIN                                                 20007660
      COMMON / CV1 / IP(12),PP(12),TMP(10)                              20007670
      DATA ENDER / 5HEND      /                                         20007680
      REWIND 7                                                          20000210
      NC=PP(3)                                                          20007690
      IF(NC.EQ.0) GO TO 20                                              20007700
      DO 10 I=1,NC                                                      20007710
      READ (5,100)  (TMP(J),J=1,8)                                      20007720
      WRITE(7,100)  (TMP(J),J=1,8)                                      20007730
  100 FORMAT (8A10)                                                     20007740
   10 CONTINUE                                                          20007750
   20 WRITE(7,100)  ENDER                                               20007760
      RETURN                                                            20007770
      END                                                               20007780
      SUBROUTINE SET(TMMAX)                                             20007790
      COMMON/TMX/TMO,EXT                                                20007800
C                                                                       20007810
C  SECOND GIVFS JOB CPU FXECUTION TIME IN 1/1000 OF A SECOND            20007820
C                                                                       20007830
      CALL SECOND(TMO)                                                  20007840
      EXT = TMMAX + TMO                                                 20007850
      RETURN                                                            20007860
      END                                                               20007870
      SUBROUTINE TABOUT(IRT)
C  LABELLED COMMON
     COMMON / CV1 / IR(12),PR(12),TMP(10)
     COMMON / CV2 / T(100,10),BO(100),BLO(10),ULO(10),CO(10)
     COMMON / CV3 / M,N,NCF,RHIT,UZ,USP,USM,EKO,MPLUS
     COMMON / CV4 / IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST
     COMMON / CV5 / SIGMA(100,4),TSIG
     COMMON / CV7 / NPHASE,NF1,CEX,IOPT,NOP,NOPS,NEWXZ
     COMMON / CV8 / NXBK,YK,NOBOL,EKBL(25)
     COMMON / CV9 / PSIGL(25),NXBL(25),XNXBL(25),BLIST(25,131)
C
      IF(NOP .GT. 1) GO TO 90
      GO TO 100
 90   CONTINUE
      IF(PR(4) .EQ. 0.0) GO TO 777
100   CONTINUE
      WRITE(6,101)
101   FORMAT(1H1,30HTABOUT - GENERAL - INFORMATION)
     IF (IRT.NE.3) CALL TIMEC
      WRITE(6,105)UZ,USP
105   FORMAT(1H0,6HUZERO=,1PE18.7,6X,5HUSP =,1PE18.7)
     IF (NEWXZ.EQ.1)
    *WRITE (6,570) (IXZ(I),XZ(I),I=1,MPLUS)
570 FORMAT (7H0XZERO // (7X,5(7H   COL ,I4,2H =,F12.4)))
     IF (IRT.EQ.3) GO TO 777
     NEWXZ=0
      WRITE(6,109)
109   FORMAT(1H0,10HSIGMA(I,J),13X,5HPHS-R,12X,6HLW-BND,12X,6HUP-BND,
    111X,7HC-SLOPE)
C
C
      DO 115 I=1,NCF
      WRITE(6,113) I, (SIGMA(I,J),J=1,4)
113   FORMAT(1H ,5X,I2,7X,4F18.5)
115   CONTINUE
C
      WRITE(6,117)TSIG
117   FORMAT(1H0,6HE(K) =,F18.6)
C
      IF(IRT.NE. 1)GO TO 145
C
     IF(NOBOL.LE.0) GO TO 145
      IT = NOBOL
      DO 131 I=1,IT
      WRITE(6,121)I,PSIGL(I),NXBL(I),XNXBL(I),EKBL(I)
121   FORMAT(1H-,6HNODE =,I4,6X,6HCOST =,F20.6,6X,8HNX-BRN =,I4,6X,
    1 7HX-BRN =,F20.6,6X,5HE(K) =,F20.6)
131    CONTINUE
145 CONTINUE
777   RETURN
      END
      SUBROUTINE TIMEC
      COMMON / CV3 / M,N,NCF,PHIT,UZ,USP,USM,EKO,MPLUS
      COMMON / CV4 / IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST
      COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
       COMMON /TSW/ NSWW
       COMMON/TMX/TMO,EXT,TITLE(4)
C
C  SECOND GIVES JOB CPU EXECUTION TIME IN 1/1000 OF A SECOND
C
       CALL SECOND(SECS)
       XX= SECS - TMO
      WRITE(6,666) XX
  666 FORMAT(12H0EXCT-TIME =,F9.3,8H SECONDS)
       IF(SECS .LT. EXT) GO TO 100
       WRITE(6,667)
  667  FORMAT(37H TIME IS UP...CYCLING TO NEXT PROBLEM)
      MNC=(-1)*NCF
 4446 WRITE(8,4448)(TITLE(I),I=1,4)
 4448 FORMAT(4A10)
      WRITE(8,4447)(IXZ(I),XZ(I),I=1,MPLUS)
 4447 FORMAT(I4,4X,F12.4)
      WRITE(8,4447)MNC,UZ
      NEWXZ=1
      CALL TABOUT (3)
      CALL BBCAV2
  100  RETURN
       END
      SUBROUTINE LP(MROWS,NCOLS,NCHGS)
      COMMON / CV1 / IP(12),RP(12),TMP(10)
      COMMON / CV2 / T(100,10),BO(100),BLO(10),UBS(10),CO(10)
      COMMON / CV4 / IX(110),X(110),IXZ(110),XZ(110),XCON(10),COST
      COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
C---------SET RHS TO INPUTM+1
      COMMON /RHS/ RHS(100)
C---------SET AJ(AS MUCH AS POSSIBLE) OVER INPUTM+1**2 FOR CORE COLUMNS
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(11000)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /DJS/ DJ(100)
C---------SET IROWTP(INPUTM+1)  NAME(INPUTM+INPUTM+1)
      COMMON /ROWTYP/ IROWTP(101) /NAMES/ NAME(600)
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /FILES/ IA1,IA2,IMAP
      COMMON /INPUT/INPUT,INPUTM,INPUTN
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      COMMON/IXX/IXX(100) /XX/ XX(100)
C-----THE B ORIGIN IS MOVED DOWN THE AJ SPACE --IGNORE SIZE
      REAL B(100)
      EQUIVALENCE (AJ,B)
C
      CALL MSSG(40HLP/LONG/5-GUB CYCLIC                              )
C-----FILE DEFINITIONS
      IA1=1
      IA2=2
      INPUT=3
      IMAP=7
      REWIND INPUT
      CALL FTNBIN(1,1,IA1)
      CALL FTNBIN(1,1,IA2)
C-----SET LENGTH OF AJ SPACE IN NWAJ
      NWAJ=11000
C
C
C
C
C
C
C
C
      ICOST=INPUTM=MROWS
      JRHS=INPUTN=NCOLS
      NBDS=NCHGS
      DO 10 I=1,NBDS
      IBDS(I)=I
 10   BOUNDS(I)=UBS(I)
C-----MAPOUT AFTER TMAX, QUIT AFTER K5 CYCLES
      TMAX=200.
      K5=200
C-----XCHECK BETWEEN CYCLES N000 TO NNNAT INCREMENTS K2
       K2 = 1
      K4=100
      K4=0
      K4=1
C-----PRINT CONTROL K3
       K3 = 0
      K3=5*7*11*13
      IF(IP(8).NE.1) K3=K3/5
      IF(IP(7).EQ.0) K3=13*7
      IF(PP(4).EQ.0.0) K3=1
C
C
C
C
C
C
C
C
C
C
C
C
C-----PROGRAM VERBS
 100  CALL SETUP
       WRITE(6,999) IROWTP
999    FORMAT(* DUMP IROWTP*/(1X,50I1))
C-----M IS NOW ACTUAL NON-GUB ROWS, L IS NO. OF GUB ROWS NWAJ IS AJ SPAC
      MPL=M+L
C-----OPTIMIZE CORE COLUMN STORAGE
      IORG=NWAJ-M*M
      NCRMAX=MIN0( 98,IORG/M)- 3
 200  CALL MAPIN(B(IORG))
 250  FORMAT(//* LP PROBLEM DATA FOR THIS RUN *
     +          /* NON-GUB ROWS * I6
     +          /*      GUB-ROWS * I6
     +          /
     +          /* LOGICALS     * I6
     +          /* TOTAL COLUMNS* I6
     +          /* MAX IN CORE  * I6
     +          /* INVERT FREQU.* I6 * CYCLES*
     +          /* MAX RUN TIME * F6 * SECONDS*
     +          /* MAX CYCLES   * I6 * ITERATIONS*
     +    ////)
      WRITE(6,250) M,L,MC,NT,NCRMAX,INVF,TMAX,K5
      IF (IORG.GE.2*M) GOTO 300
      CALL ERROR(40HLP--INSUFFICIENT  SPACE STATED IN NWAJ      )
      CALL ESCAPE(B(IORG))
 300  CALL INVERT(B(IORG))
 400  CALL PRIMAL(B(IORG))
      ITNINV=0
      CALL INVERT(B(IORG))
      IPI=IORG+IPT-1
      DO 500 J=1,NT
      CALL IN(J,AJ,0)
 500  DJ(J)=DOT(M,B(IPT),AJ)
      IPI=IPI-IORG+1
      WRITE(6,501) (J,DJ(J),NAME(J),J=1,NT)
 501  FORMAT(*0DJ VALUES FOR FINAL SOLUTION COLUMNS*/(I10,E12.4,I10))
C-----END PHASE 2, OR UNBOUNDED OR NO FEASIBLE SOLUTION
 800  CONTINUE
      CALL MAPOUT(B(IORG))
      NVAR=INPUTM+NCHGS
      DO 900 I=1,NVAR
      IX(I)=IXX(I)
  900 X(I)=XX(I)
      COST=-BETA(ICOST)
 7776 RETURN
      END
      FUNCTION BOUND(J)                                                 20009880
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS                        20009890
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI                 20009900
      COMMON /NAMES/ NAME(100)                                          20009910
C-----PICKS UP BOUND FROM PACKED LIST,EITHER  FINITE OR  10**35         20009920
      BOUND=1.E70                                                       20009930
      IF(J.GT.NT) RETURN                                                20009940
      IB=NAME(J)/100000                                                 20009950
       IF(IB.LE.0.OR.IB.GT.NBDS) RETURN                                 20009960
      BOUND = BOUNDS(IB)                                                20009970
      RETURN                                                            20009980
      END                                                               20009990
      SUBROUTINE COLUMN(JCOL,B)
C-----GUB VERSION  APRIL 20-71
      COMMON  /MOVES/ THETA,BNDJ,DMAX,PRMLER,DUALER
      COMMON /STATE/ JPOS,IROW,JKOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      COMMON /T/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /NAMES/ NAME(100)
      COMMON /DJS/ DJ(100)
      LOGICAL  BASIC,ATBND,NULL,KEY
      REAL B(1)
      KEY(I)=MOD(NAME(I),10).EQ.4
      NPKT(J)=MOD(NAME(J),100000)/10
C
C-----CHECKS COLS IN CORE, IF NONE GETS SOME, IF SOME FINDS BEST
      NTRY = 1+NTRY
      MAXTRY=NCRMAX
      IF( NTRY.GT. MAXTRY)  GOTO 1
      IF( JNCORE.NE.0)      GOTO 5
C-----CHECK FOR MORE COLUMNS ON DISC
 1    CALL DISC(B)
      NTRY = 0
      NDJST=NDJS
      IF( JNCORE.EQ.0)      GOTO 100
C
C-----RE-PRICE  VECTORS IN CORE
 5    JORG = 1
C-----NO PRICING IF REJECTS OR JUST PRICED DISC
      IF(NREJ.NE.0  .OR.  NTRY.EQ.0) GOTO 50
C-----PRICE OUT COLUMN
      DO 40  J= 1,JNCORE
      DJ(J)=DOT(M,B(IPI),AJ(JORG))
 40   JORG = JORG+M
C
C-----NOW FIND BEST COLUMN IN CORE, NON-BASIC OR BOUNDED
 50   DMAX=0
      NDJS=0
      JPKTO=0
      PIKEY=0.
      DO  60  J=1,JNCORE
      IF(JAREJ(J).EQ.1) GOTO 60
      JPOS = JA(J)
      JTYPE=MOD(NAME(JPOS),10)
      IF(JTYPE.EQ.2) GOTO 60
      IF(JTYPE.EQ.4) GOTO 60
      IF(JTYPE.EQ.0) GOTO 60
      JPKT=NPKT(JPOS)
      IF(JPKT.EQ.0) GOTO 55
      IF(JPKT.EQ.JPKTO) GOTO 55
      JKEY=KEYFND(JPKT)
C-----NEW PACKET STARTED, FINE KEY AND KEY PRICE
      IF(JKEY.EQ.0) GOTO 60
      PIKEY=DJ(JKEY)
      JPKTO=JPKT
 55   D=DJ(J)
```

```
      IF(JTYPE.EQ.3) D=-DJ(J)
      IF(JPKT.NE.0) D=D-PIKEY
      IF(D.LT.-ZERO)  NDJS=1+NDJS
      IF(D.GE. DMAX)  GOTO 60
      DMAX =D
      JCOL = J
 60   CONTINUE
      NCORE=JNCORE
C-----RESTORE COUNT OF NDJS FROM CHECK IF JUST DONE
      IF(NTRY.EQ.0 .AND. NT.GT.NCRMAX) NDJS=NDJST
      IF( DMAX.LT.-DJTOL)  GOTO 70
C-----CURRENT COLS NO-GOOD,  QUIT IF THESE ARE BEST
      IF( NTRY.EQ.0)       GOTO 100
      GOTO 1
C
C-----RETURN WITH COLUMN INDEX
  70  RETURN
C
C-----NO GOOD COLS,  OPTIMUM
 100  JCOL=0
C-----SAVE OLD COLUMNS
      JNCORE=NCORE
      RETURN
C
      END
      SUBROUTINE DISC(B)
C     REVISED  10/71
C-----CHECKS DISC FOR COLUMNS, ACCEPTING 1/NBCH, IF NOT ALL IN CORE.
C     RETURNS  JNCORE COLUMNS AND PRICES,  OR JNCORE=0
C-----PACKETS CAN IN BE INTER-MIXED WITH SOME LOSS OF EFFICIENCY
C     DUE TO MULTIPLE KEY SEARCHES
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPI
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      COMMON /PARAMS/ IMAX,ITNINV,INVF,K1,K2,K3,K4,ITNCHK
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /DJS/ DJ(100)
      COMMON /NAMES/ NAME(100)
      INTEGER PKT,PKTO
      REAL B(1)
      LOGICAL BASIC,ATBND ,NULL,CHEK
      NPKT(J)=MOD(NAME(J),100000)/10
      NULL(I)=MOD(NAME(I),10).EQ.0
C
C-----CHECK FOR AN INVERT (  ITRN.GE.ITNINV)
      CALL INVERT(B)
C
C-----ALL IN CORE,  NCRMAX SET IN LP
      IF( NT.LT.NCRMAX) GOTO 200
C-----ACCEPT 1 COL/NBCH COLS,  BEST AT IORG, NEW AT JORG, (ICOL,JCOL)
      NBCH=NT/NCRMAX/4+1
      NDJS=PKTO=JNCORE=0
      JORG=0
      JCOL=1
      IORG=M
      ICOL=2
      NCORE=2
      CALL  INPOS(JNT)
      NBCHS = (NT+NBCH-1)/NBCH
C
      DO 1000  JBCH  =1, NBCHS
      DJOLD = 1.E35
      DO 100   JFBCH= 1,NBCH
C-----BATCH CYCLE,  NEXT COLUMN  JNT
      JNT= 1+MOD(JNT,NT)
      JTYPE =  MOD(NAME(JNT),10)
C-----SKIP NULL, BASIC  OR KEY COLUMNS
      IF(JTYPE.EQ.2)  GOTO 100
      IF(JTYPE.EQ.4)  GOTO 100
      IF(JTYPE.EQ.0)  GOTO 100
C
C-----IF IN A GUB PACKET, GET KEY
      PKT= NPKT(JNT)
      IF( PKT.EQ.0    )  GOTO   20
      IF( PKT.EQ.PKTO)  GOTO   20
C-----USE AN UNUSED KEY SLOT
      JKEY=KEYFND(0)
      IF(JKEY.NE.0) GOTO 15
 10   NCORE=1+NCORE
      JKEY=NCORE
 15   KORG= M*JKEY-M                                                20011390
      CALL INPCKD(KEYS(PKT)/100,AJ(KORG+1),JKEY)                    20011400
      DJ(JKEY)= DOTS(M,B(IPI),AJ(KORG+1) )                          20011410
      PKTO=PKT                                                      20011420
C                                                                   20011430
C-----NOW GET COLUMN  AND DJ.                                        20011440
 20   CONTINUE                                                      20011450
      CALL IN(JNT, AJ(JORG+1), JCOL)                                20011460
      DJ(JCOL)= DOTS(M,B(IPI),AJ(JORG+1) )                          20011470
C                                                                   20011480
C-----CORRECT FOR PACKET AND BOUND EFFECTS                           20011490
      DJNEW = DJ(JCOL)                                              20011500
      IF(PKT.NE.0)  DJNEW = DJNEW - DJ(JKEY)                        20011510
      IF(JTYPE.EQ.3)DJNEW =-DJNEW                                   20011520
      IF( DJNEW.LT.-ZERO)   NDJS=1+NDJS                             20011530
C                                                                   20011540
C-----SELECTION STAGE INTERCHANGE BEST FOR NEW                       20011550
      IF(DJNEW.GE.DJOLD)  GOTO 100                                  20011560
      DJOLD=DJNEW                                                   20011570
      I=ICOL                                                        20011580
        ICOL=JCOL                                                   20011590
             JCOL=I                                                 20011600
      I=IORG                                                        20011610
        IORG=JORG                                                   20011620
             JORG=I                                                 20011630
 100  CONTINUE                                                      20011640
C                                                                   20011650
      IF( DJOLD.GT.-DJTOL) GOTO  999                                20011660
C                                                                   20011670
C-----PRESERVE THE BEST                                              20011680
      JNCORE=1+JNCORE                                               20011690
       NCORE=1+NCORE                                                20011700
      ICOL  = NCORE                                                 20011710
      IORG  = M*ICOL-M                                              20011720
 999  IF(  NCORE.GE. NCRMAX) GOTO 110                               20011730
 1000 CONTINUE                                                      20011740
 110  CONTINUE                                                      20011750
      IF(JNCORE.NE.0) JNCORE=NCORE-1                                20011760
      GOTO 500                                                      20011770
C                                                                   20011780
C-----ALL IN CORE CASE,  READ AND PRICE .                            20011790
 200  IF( JNCORE.EQ.0 ) GOTO250                                     20011800
      JNCORE=0                                                      20011810
      GOTO 500                                                      20011820
C                                                                   20011830
 250  CONTINUE                                                      20011840
      JORG=0                                                        20011850
      DO 300 JNT=1,NT                                               20011860
      CALL IN(JNT,AJ( JORG+1),JNCORE+1)                             20011870
      IF( NULL(JNT) )GOTO 300                                       20011880
      JNCORE=1+JNCORE                                               20011890
      DJ(JNCORE) = DOTS(M,B(IPI), AJ(JORG+1))                       20011900
      JORG = M+JORG                                                 20011910
 300  CONTINUE                                                      20011920
      GOTO 500                                                      20011930
C                                                                   20011940
C-----DIAGNOSTICS  IF K3*23                                          20011950
 500  CONTINUE                                                      20011960
```

```
      IF( MOD(K3,23).NE.0 )  RETURN
      WRITE(6,501)     (JA(J), DJ(J), J=1,JNCORE )
  501 FORMAT(* DISC-PROVIDED*/(8( I5, E10.2 )) )
      RETURN
      END
      FUNCTION DOT(M,X,Y)
C-----INNER PRODUCT OF X AND Y
      DOUBLE PRECISION  SUM
      REAL  X(1),Y(1)
      SUM=0.0
      DO  100  I=1,M
      IF(Y(I).EQ.0.0) GOTO 100
      SUM=SUM+X(I)*Y(I)
 100  CONTINUE
      DOT = SUM
      RETURN
C
C-----SINGLE PRECISION VERSION FOR SPEED
      ENTRY DOTS
      DOT=0.0
      DO 200 I=1,M
 200  DOT=DOT+X(I)*Y(I)
      RETURN
      END
      SUBROUTINE ESCAPE(B)
C------GUB VERSION APRIL/71
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NORMAX,NSCAN
      COMMON /T/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPI
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /NAMES/ NAME(100)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /DJS/ DJ(100)
      DATA AALPHA/5HALPHA/,ABETA/4HBETA/,AGAMMA/5HGAMMA/,API/2HPI/
      DATA  AJAREJ/5HJAREJ/,AJA/2HJA/,ANAME/4HNAME/,ABASIS/5HBASIS/
      DATA ADELTA/5HDELTA/,ADJ/2HDJ/, AKEY/3HKEY/
      COMMON /RHS/ RHS(100)
      NPKT(J)=MOD(NAME(J),100000)/10
      CALL MESSG(40HESCAPED AFTER DUMP OF LP SYSTEM          )
      WRITE(6,3)  ANAME
      WRITE(6,2) (NAME(J),J=1,NT)
      WRITE(6,3)  ABASIS
      WRITE(6,2) (IBASIS(J),J=1,M)
      WRITE(6,3)AKEY
      WRITE(6,2)(KEYS(I),I=1,L)
      WRITE(6,3)  AJA
      WRITE(6,2) (JA(J),J=1,JNCORE)
      WRITE(6,3)  AJAREJ
      WRITE(6,2) (JAREJ(J),J=1,JNCORE)
      WRITE(6,3)  AALPHA
      WRITE(6,1) (ALPHA(J),J=1,MPL)
      WRITE(6,3)  ABETA
      WRITE(6,1) (BETA (J),J=1,MPL)
      WRITE(6,3)  AGAMMA
      WRITE(6,1) (GAMMA(J),J=1,M)
      WRITE(6,3) ADELTA
      WRITE(6,1)(DELTA(J),J=1,M)
      WRITE(6,3)  ADJ
      WRITE(6,1) (DJ    (J),J=1,JNCORE)
 1    FORMAT(1H ,10E12.5)
 2    FORMAT(1H ,10I12)
 3    FORMAT(1H0,A10)
      CALL MAPOUT(B)
C-----CAUSE A DUMP
      I=0
      WRITE(I) I
      RETURN
      END
      SUBROUTINE EXITS
      COMMON / CV7 / NPHASE,NF1,CFX,IOPT,NOP,NOPS,NEWXZ
C-----PRIMAL CALLS THESE ENTRY POINTS AT THE END OF EACH PHASE----------
C-----------------------------------------------------------------------
      ENTRY OPT1
      NPHASE=1
      RETURN
C-----------------------------------------------------------------------
      ENTRY OPT2
      NPHASE=2
       NF1 = 1
      CALL STATUS(40HPRIMAL--END OF PHASE 2--OPTIMAL         )
      RETURN
C-----------------------------------------------------------------------
      ENTRY UNBND
      NPHASE=4
      CALL STATUS(40HPRIMAL--UNBOUNDED SOLUTION              )
      RETURN
C-----------------------------------------------------------------------
      ENTRY NOFEAS
      NPHASE = 5
      CALL STATUS(40HPRIMAL--NO FEASIBLE SOLUTION            )
      RETURN
C-----------------------------------------------------------------------
      END
      SUBROUTINE FEASCH(B)
C------GUR VERSION APRIL/71
C-----GIVEN CURRENT INVERSE B, RHS, KEY AND BOUNDS IN GAMMA
C-----COMPUTES CURRENT SOLUTION BETA, NO. OF INFEASIBLES NPIF.
C-----IF BETA INFEAS, ADDS ARTIFICIALS AND REVERTS TO PHASE-1
      COMMON /PARAMS/ IMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS
      COMMON /NAMES/ NAME(100)
      COMMON /RHS/ RHS(100)
      LOGICAL KEY
      REAL B(1)
      NPKT(I)=MOD(NAME(I),100000)/1)
      ITP(J)=MOD(IBASIS(J),100)
      KEY=.FALSE.
      DO 10 I=1,M
 10   DELTA(I)=0.0
      DELTA(M)=1.0
C-----COMPUTE EFFECTIVE RHS IN GAMMA USING  KEY + BOUNDS ALREADY  THERE
      DO 20 I=1,M
 20   GAMMA(I)=+GAMMA(I)+RHS(I)
C-----CYCLE ELEMENTS OF SOLUTION
      NPIF=0
      SUMIF=0.
      BNDJ=1.E70
      IORG=1
      MM1=M-1
      DO 100 I=1,MM1
      SUM = DOT(M, B(IORG), GAMMA )
      BETA(I)=SUM
      IORG= M+IORG
      JOUT= IBASIS(I)/100
      IF(NBDS.EQ.0) GOTO 50
C-----CHECK XJOUT LESS THAN BOUND
      BNDJ = BOUND(JOUT)
      IF( SUM.LE.BNDJ+ZERO) GOTO 50
C-----BOUND VIOLATED, REMOVE BOUND AND TREAT AS INFEAS XJOUT
      IF(MOD(K3,13).EQ.0) WRITE(6,101) I,JOUT,SUM,BNDJ
      CALL SETBND (JOUT)
      BETA(I)=BETA(I)-BNDJ
      GOTO 60
C-----CHECK XJOUT POSITIVE FOR NON-FREE ROWS
 50   IF( ITP   (I).EQ.3 )  GOTO 100
      IF( BETA(I).GE.-ZERO) GOTO 100
C-----INFEASIBLE,  ADD ARTIFICIAL TO KILL ERROR AND PIVOT IN
      IF(MOD(K3,13).EQ.0) WRITE(6,101) I,JOUT,SUM,BNDJ
 55   CONTINUE
      CALL SETBNB(-JOUT)
      BETA(I)= -BETA(I)
 60   NPIF=1+NPIF
      JPKT=0
      IF(JOUT.LE.NT) JPKT=NPKT(JOUT)
      IF(JPKT.NE.0) KEYS(JPKT)=KEYS(JPKT)-1
      SUMIF=SUMIF+BETA(I)
      IBASIS(I)=100*(100+JOUT+NT)+MOD(IBASIS(I),100)
      DELTA(I)= -1.
      IF(JOUT.GT.NT) DELTA(M)=2.
      CALL PIVOT(I,B,DELTA)
      DELTA(I)=  0.
      DELTA(M)=1.
      IF(KEY) GOTO 150
 100  CONTINUE
 101  FORMAT(* INFEAS--ROW*I5*  COL*I5*  VALUE*E12.4*  BOUND*E12.4)
C-----CONSTRUCT COMPLETE SOLUTION FOR KEYS  IN BETA(M+1).....
      IF(L.EQ.0) GOTO 151
      BNDJ=1.E70
      KEY=.TRUE.
      DO 150 K=1,L
C-----CHECK PACKET HAS BASIC COLS
      SUM= 0.0
      NB =  MOD (KEYS(K),100)
      IF(NB .EQ.0 )  GO TO  115
C-----SUM BASIC COL VALUES IN PACKET K
      DO 110 I=1,M
      JPOS = IBASIS(I)/100
      IF(NPKT(JPOS).EQ.K) SUM=SUM+BETA(I)
 110  CONTINUE
 115  BETA(M+K) =  RHS(M+K)-SUM
      IF(BETA(M+K).GE.-ZERO) GOTO 150
C-----INFEASIBLE GUB, MUST BE ESSENTIAL, CHANGE TO BASIC ROW I
      JOUT=KEYS(K)/100
      IF(MOD(K3,13).EQ.0) WRITE(6,111) K,JOUT,BETA(M+K),BNDJ
 111  FORMAT(* INFEAS--GUB*I5*  COL*I5*  VALUE*E12.4*  BOUND*E12.4)
      CALL KEYCH(JOUT,I,B)
      GOTO 55
 150  CONTINUE
 151  CONTINUE
C-----TOTAL INFEASIBILITY
      BETA(M)=0.
      DO 160 I=1,MM1
      IF(IBASIS(I)/100.LE.NT) GOTO 160
      BETA(M)=BETA(M)-BETA(I)
 160  CONTINUE
C-----INDICATE PHASE 1
      IF(ABS(BETA(M)).LT.CTOL) GOTO 200
      IPHASE = 1
      IF(MOD(K3,13).EQ.0) WRITE(6,199) SUMIF,BETA(M)
 199  FORMAT(* TOTAL INFEASIBILITY*E12.4* PHASE1 COST*E12.4)
C-----ADJUST COST ROW           IC AND  ORIGIN IPI
 200  IC = ICOST
      IF(IPHASE.EQ.1)   IC = M
      IPI=1+M*IC-M
C-----PHASE 1 COST IS EQUALITY ZERO IN PHASE 2
      IBASIS(M)=100*(IBASIS(M)/100)
      IF(IPHASE.EQ.1) IBASIS(M)=IBASIS(M)+3
      RETURN
      END
      SUBROUTINE INVERT(B)
C------SUB VERSION APRIL/71
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DEPTOL
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,ITNCHK
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPI
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /NAMES/ NAME(100)
      COMMON /RHS/ RHS(100)
      REAL B(1)
      INTEGER PKT,PKTO
      LOGICAL BASIC,ATBND
      ITP(J)=MOD(IBASIS(J),100)
      NPKT(J)=MOD(NAME(J),100000)/10
C-----INVERTS CURRENT BASIS VECTORS AND ADJUSTS FOR BOUNDS
      IF(ITRN.LT.IT NINV) RETURN
      CALL MESSG(40HINVERT                                    )
C-----ITERATION OF NEXT INVERT
      ITNINV=ITRN+INVF
C
      IF(L.EQ.0) GOTO 15
C-----COUNT MISSING KEYS TO NKM AND CLEAR BASIC COUNT
      NKM=0
 0    DO 5 I=1,L
      K=KEYS(I)/100
      IF(K.EQ.0) NKM=NKM+1
      KEYS(I)=100*K
 5    CONTINUE
      IF(NKM.EQ.0) GOTO 15
C-----FIRST SCAN BASIC COLS FOR KEY CANDIDATES
      JTAG=2
 6    CONTINUE
      DO 10 J=MC,NT
      K=NPKT(J)
      IF(K.EQ.0) GOTO 10
      JTYPE=MOD(NAME(J),10)
      IF(JTYPE.NE.JTAG) GOTO 10
      IF(KEYS(K)/100.NE.0) GOTO 10
C-----SET COLUMN J KEY IN PACKET K AND REDUCE COUNT NKM
      KEYS(K)=100*J
      CALL SETKEY(J)
      NKM=NKM-1
 10   CONTINUE
      IF(NKM.EQ.0) GOTO 15
      IF(JTAG.NE.2) GOTO 11
C-----NOW EXAMINE FREE COLUMNS
      JTAG=1
      GOTO 6
 11   CONTINUE
      CALL ERROR(40HINVERT--SEVERAL KEYS UNMARKED-DATA ERR.      )
      CALL ESCAPE(B)
C
C-----NULL BASIS EXCEPT FOR FREE ROWS SAVING TYPES
```

```
 15   CONTINUE
      IORG=0
      DO 20 I=1,M
      GAMMA(I)=0.0
      ITYPE=MOD(IBASIS(I),100)
      IF(ITYPE.NE.3) IBASIS(I)=ITYPE
C-----SET UP UNIT BASIS AND ZERO RHS
      DO 19 J=1,M
 19   B(IORG+J)=0.
      B(IORG+I)=1.0
 20   IORG=IORG+M
C-----RESTORE PHASE1 LOGICAL
      IBASIS(M)=100*MC+ITYPE
C
C-----CYCLE COLUMN NAMES, KEY COLUMNS TO KORG, OTHERS TO JORG
      JORG=M*JNCORE
      KORG=JORG+M
C-----PKT IS CURRENT GUP PACKET, PKTO IS PACKET OF LAST KEY
      PKTO=0
      DO 200 JNT=1,NT
      JTYPE= MOD( NAME(JNT),10)
      IF(JTYPE.LE.1) GOTO 200
C-----GET THIS BASIC/BOUNDED/KEY COL TO CORE
      JPOS=JNT
 30   CALL IN(JPOS,AJ(JORG+1),JNCORE+1)
      PKT=NPKT(JNT)
      IF(JTYPE.GE.3) GOTO 150
C-----BASIC COLUMN, IS KEY NEEDED
      IF(PKT.EQ.0) GOTO 120
      IF(PKT.EQ.PKTO) GOTO 100
C-----GET KEY AND RECORD
      CALL INPCKD(KEYS(PKT)/100,AJ(KORG+1),JNCORE+2)
      PKTO=PKT
C-----REMOVE KEY COMPONENT FROM COL
 100  DO 110 I=1,M
 110  AJ(JORG+I)=AJ(JORG+I)-AJ(KORG+I)
C-----TRANSFORM TO CURRENT BASIS
 120  IORG=1
      DO 130 I=1,M
      ALPHA(I)=DOT(M,B(IORG),AJ(JORG+1))
 130  IORG=IORG+M
C-----FIND BEST ROW TO PIVOT
      IROW=0
      CALL PIVOT(IROW,B,ALPHA)
      IF( IROW.EQ.0) GOTO 200
C-----INCREASE COUNT OF BASIC COLS IN PACKET
      IF(PKT.NE.0) KEYS(PKT)=KEYS(PKT)+1
      GOTO 200
C
C-----PICK UP BOUND OR RHS OF PACKET
 150  IF(PKT.NE.0) GOTO 155
      BNDJ=BOUND(JPOS)
      GOTO 156
 155  BNDJ=RHS(PKT+M)
 156  DO 160 J=1,M
 160  GAMMA(J) = GAMMA(J)- AJ(JORG+J)*BNDJ
 200  CONTINUE
C
```

```
C-----COMPLETE BASIS WITH ARTIFICIALS
C-----COUNT LOGICALS IN JL
      JL=0
      DO  210 I=1,M
      IF(MOD(IBASIS(I),100).NE.0) JL=JL+1
      IF( IBASIS(I)/100.NE.0 ) GOTO 210
      IF(MOD(IBASIS(I),100) .NE.1) GOTO 205
C-----MAKE A LOGICAL BASIC INSTEAD
      IBASIS(I)=100*JL+IBASIS(I)
      GOTO 210
 205  CONTINUE
      IBASIS(I)=100*(I+NT)+IBASIS(I)
 210  CONTINUE
C-----ADD ARTIFICIALS NEEDED
      DO 220 I=1,M
 220  DELTA(I)=0.0
      DELTA(M)=1.0
      DO 240 I=1,M
      IF(IBASIS(I)/100.LE.NT) GOTO 240
C-----ONE NEEDED ROW I
      DELTA(I)=1.0
      IORG=1
      DO 230 J=1,M
      ALPHA(J)= DOT(M,B(IORG),DELTA)
 230  IORG=IORG+M
      DELTA(I)=0.0
      CALL PIVOT(I,B,ALPHA)
 240  CONTINUE
C
C-----NOW USE B AND GAMMA TO GET SOLUTION TO BETA
      CALL FEASCH(B)
      RETURN
      END
      SUBROUTINE IO(KOL, ALPHA, NAME)
C-----GUB VERSION--APRIL-20-1971
C-----WRITES TWO FILES OF A MATRIX TO DISC   IN  STRAIGHT OR PACKED FORM
      COMMON /FILES/ IA1,IA2,IMAP
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /ROWTYP/ IROWTP(101)
      COMMON /LIMS/ MAXTPY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      DIMENSION ALPHA(1),ID(100),D(100)
      COMMON /B/ ALPHB(100)
      DATA ZERO/1.E-10/
C
      ENTRY  OUT
C----------------TO DISC FILES IA1/IA2
C
      IF (KOL.GT.1)  GOTO 10
      REWIND IA1
      REWIND IA2
      KOL1=KOL2=0
C
C-----STRIP GUBS AND PACK  FOR TWO FILES
 10   J=K=0
      DO 20  I=1, M
      IF( IROWTP(I).EQ.4 )  GOTO  20
      J=J+1
      ALPHB(J)=ALPHA(I)
      IF(ABS(ALPHA(I)).LT.ZERO) GOTO 20
      K=K+1
      ID(K)=J
      D(K)=ALPHA(I)
 20   CONTINUE
      NAME=KOL
      WRITE(IA1) KOL,NAME,(ALPHB(I),I=1,J)
      WRITE(IA2) KOL,NAME,K,(ID(I),D(I),I=1,K)
      RETURN
C
C
      ENTRY  IN
C----------------FOR NORMAL COLUMNS FROM DISC IA1
      DO 100 JNT=1,NT
      IF(MOD(KOL1,NT).NE.0) GOTO 110
 99   REWIND IA1
      REWIND IA2
      NSCAN =1+NSCAN
 110  READ(IA1) KOL1,NAAM,(ALPHA(I),I=1,M)
      IF(KOL.LT.KOL1) GOTO 99
      IF(KOL.EQ.KOL1) GOTO 101
 100  CONTINUE
      GOTO 300
 101  CONTINUE
C-----UPDATE RECORDS  AND TRACK  DISC LOCATION IN KOL
 120  CONTINUE
      JCOL=NAME
      JA(JCOL)=KOL
      JAK(JCOL)= NAAM
      JAREJ(JCOL)=0
      RETURN
```

```
C
C
      ENTRY INPOS
C-----TO GET THE INPUT FILE POSITION
      KOL=KOL1
      RETURN
C
C
      ENTRY INPCKD
C-------------AUXILIARY FILE FOR KEYS
      DO 200 JNT=1,NT
      IF( MOD(KOL2,NT).NE.0) GOTO 199
  195 REWIND IA2
  199 READ(IA2) KOL2,NAAM,K,(ID(I),D(I),I=1,K)
      IF(KOL.LT.KOL2) GOTO 195
      IF(KOL2.EQ.KOL) GOTO 201
  200 CONTINUE
      GOTO 300
  201 CONTINUE
C-----UNPACK D TO ALPHA
      DO 210 I=1,M
  210 ALPHA(I)=0.
      IF(K.EQ.0) GOTO 120
      DO 220 I=1,K
      J=ID(I)
  220 ALPHA(J)=D(I)
      GOTO 120
C
C-----TROUBLE
  300 CALL ERROR (40HIO--COLUMN NOT LOCATED IN NT READS          )
      CALL ESCAPE( 8 )
      END
      SUBROUTINE KEYCH(JCOL,IROW,B)
C------GUB VERSION APRIL/71
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /NAMES/ NAME(100)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      REAL B(1)
      NPKT(J)=MOD(NAME(J),100000)/10
      ITP(J)=MOD(IBASIS(J),100)
C
C-----INTERCHANGE KEY WITH FIRST BASIC COL IN KEYS PACKET
C     RETURNING ROW OF NEW BASIC COL (OLD KEY)
      JCOLPK= NPKT(JCOL)
C-----FIRST BASIC COL
      DO 10 I=1,M
      JKEY  = IBASIS(I)/100
      IF(JKEY.GT.NT) GOTO 10
      IF(NPKT(JKEY).EQ.JCOLPK) GOTO 20
 10   CONTINUE
      CALL ERROR(40HKEYCH--ESSENTIAL PACKET NO BASIC COL          )
      CALL ESCAPE
C
C-----RE-DIFFERENCE BASIS INVERSE TO MAKE JKEY KEY
 20   IROW = I
      JORG = M*IROW-M
      DO 30 J=1,M
 30   B(JORG+J) =-B(JORG+J)
      IORG = 0
      DO 50 I=1,M
      IF(I.EQ.IROW) GOTO 50
      IB=IBASIS(I)/100
      IF(IB.GT.NT) GOTO 50
      IF(NPKT(IB).NE.JCOLPK) GOTO 50
      DO 40 J=1,M
 40   B(JORG+J)=B(JORG+J)-B(IORG+J)
 50   IORG=IORG+M
C
C-----NEW KEY IS NOW JKEY
      CALL SETKEY(JKEY)
      KEYS(JCOLPK)= 100*JKEY+ MOD(KEYS(JCOLPK),100)
C-----COL JCOL IS NOW BASIC
      CALL SETBNB(JCOL)
      IBASIS(IROW)=100*JCOL+ITP(IROW)
C-----REARRANGE SOLUTION
      MPK=M+JCOLPK
      SUM=ALPHA(IROW)
          ALPHA(IROW)=ALPHA(MPK)
                      ALPHA(MPK)=SUM
      SUM= BETA(IROW)
          BETA(IROW)=BETA(MPK)
                      BETA(MPK)=SUM
      RETURN
      END
      FUNCTION KEYFND(PKT)
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /NAMES/ NAME(100)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      INTEGER PKT
      NPKT(I)=MOD(NAME(I),100000)/10
C
C-----GIVEN PACKET NO. PKT, FIND ITS KEY IN CORE
      IF(PKT.EQ.0) GOTO 100
      KEY=KEYS(PKT)/100
      DO  20 K=1,JNCORE
      IF(JA(K).EQ.KEY) GOTO 30
 20   CONTINUE
      KEYFND=0
      RETURN
 30   KEYFND=K
      RETURN
C
C-----FIND THE FIRST KEY WITH NO COLUMNS IN CORE FOR CKECK
 100  DO 130 K=1,JNCORE
      JAK=JA(K)
      JTYPE=MOD(NAME(JAK),10)
      IF(JTYPE.NE.4) GOTO 130
      JPKT=NPKT(JAK)
      DO 120 J=1,JNCORE
      JAJ=JA(J)
      JTYPE=MOD(NAME(JAJ),10)
      IF(JTYPE.EQ.4) GOTO 120
      IF(JPKT.EQ.NPKT(JAJ)) GOTO 130
 120  CONTINUE
      GOTO 30
 130  CONTINUE
      KEYFND=0
      RETURN
      END
      SUBROUTINE MAPIN(B)                                               20017290
C-----GUB VERSION  APRIL 20/71                                          20017300
C-----ADDS SPECS FOR BOUND/BASIC/NULL/KEY VARIABLES AND INVERSE IF PRESE20017310
C-----OPTIONALLY CALLED BEFORE INVERT                                   20017320
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS            20017330
      COMMON /BASIS/ IBASIS(101),KEYS(101)                              20017340
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101) 20017350
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI                 20017360
      COMMON /FILES/ IA1,IA2,IMAP                                       20017370
      COMMON /INPUT/INPUT,INPUTM,INPUTN                                 20017380
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL                20017390
      COMMON /NAMES/ NAME(100)                                          20017400
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS                        20017410
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5                   20017420
      REAL B(1)                                                         20017430
      DIMENSION CARD(8)                                                 20017440
      INTEGER NAMES(5)                                                  20017450
      INTEGER PKT                                                       20017460
      REAL NULL,KEE,INVERS                                              20017470
      DATA  BASIC/5HBASIC/,ATBND/5HATBND/,ENDER/5HEND  /,ROWS/4HROWS/   20017480
     +      ,NULL/4HNULL/, KEE/3HKEY/,INVERS/5HINVER/                   20017490
     +       ,REWIND/6HREWIND/                                          20017500
      ITP(J)=MOD(IBASIS(J),100)                                         20017510
      NPKT(I)= MOD(NAME(I),100000)/10                                   20017520
C-----SETS BASIC COLUMNS AND LOGICALS                                   20017530
      CALL MESSG(40HMAPIN                                    )          20017540
      REWIND IMAP                                                       20017550
 10   READ(IMAP ,11) TYPE1,TYPE2,(NAMES(J),J=1,4)                       20017560
 11   FORMAT(2A5,4I10)                                                  20017570
      IF(MOD(K3,3).EQ.0) WRITE(6,12) TYPE1,TYPE2,(NAMES(J),J=1,4)       20017580
 12   FORMAT(X,2A5,4I10)                                                20017590
      IF(TYPE1.EQ.BASIC)  GOTO 30                                       20017600
      IF(TYPE1.EQ.KEE)  GOTO  50                                        20017610
      IF(TYPE1.EQ.ATBND) GOTO 15                                        20017620
      IF(TYPE1.EQ.NULL)  GOTO 80                                        20017630
      IF(TYPE1.EQ.INVERS) GOTO 95                                       20017640
      IF(TYPE1.EQ.ENDER)  RETURN                                        20017650
      CALL ERROR(40HMAPIN--UNRECOGNIZED TYPE CARD IN DATA        )      20017660
      RETURN                                                            20017670
C                                                                       20017680
C-----ADD AT BOUND COLUMN SPECS                                         20017690
 15   DO 20 J=1,4                                                       20017700
      ID=NAMES(J)                                                       20017710
      IF(ID.EQ.0) GOTO 20                                               20017720
      ID=ID+MC                                                          20017730
      BNDJ=BOUND(ID)                                                    20017740
      IF(BNDJ.LT.1.E8)  GOTO 19                                         20017750
      CALL ERROR(40HMAPIN--ATBND COLUMN NOT BOUNDED IBDS/BDS     )      20017760
      CALL DUMP(IBDS(1),IBDS(NBDS),2,BOUNDS(1),BOUNDS(NBDS),1)          20017770
      GOTO 20                                                           20017780
 19   CONTINUE                                                          20017790
      CALL SETBND(ID)                                                   20017800
 20   CONTINUE                                                          20017810
      GOTO 10                                                           20017820
C                                                                       20017830
C-----BASIC COLUMNS ADDED                                               20017840
 30   IF(TYPE2.EQ.ROWS) GOTO 60                                         20017850
```

```
      DO 40 J=1,4
      ID=NAMES(J)
      IF(ID.EQ.0) GOTO 40
      ID=ID+MC
      CALL SETRNB(ID)
 40   CONTINUE
      GOTO 10
C
C-----ENTER KEY COLUMNS IF A GUB PROBLEM
 50   IF(L.EQ.0) GOTO 30
      DO 55 I=1,4
      ID=NAMES(I)
      IF ( ID.EQ.0 ) GOTO  10
      ID=ID+MC
      PKT=NPKT(ID)
      IF(PKT.EQ.0) GOTO 55
      JOUT=KEYS(PKT)/100
      IF(JOUT.NE.0) CALL SETKEY(-JOUT)
      CALL SETKEY(ID)
      KEYS(PKT)=100*ID
 55   CONTINUE
      GOTO 10
C
C-----BASIC ROW-COL DATA FOR ENTRY OF ROW LOGICALS
 60   DO 70 J=1,4
      ID=NAMES(J)
      IF(ID.EQ.0)GOTO 70
      CALL SETRNB(ID)
 70   CONTINUE
      GOTO 10
C
C-----SET NULL COLUMNS
 80   DO 90 J=1,4
      ID=NAMES(J)
      IF(ID.EQ.0) GOTO 90
      ID=ID+MC
      CALL SETNNN(ID)
 90   CONTINUE
      GOTO 10
C
C-----CHECK FOR INVERSE AT END OF INPUT TAPE OR SKIP
 95   MM=M*M
      READ(INPUT) (B(J),J=1,MM)
      IF(ENDFILE INPUT) 10,96
 96   READ(INPUT) (IBASIS(J),BETA(J),J=1,M)
      IF(ENDFILE INPUT) 10,97
 97   IF(L.NE.0) READ(INPUT) (KEYS(J),BETA(J+M),J=1,L)
      IF(ENDFILE INPUT) 10,98
C-----SUCCESSFULL, SUPPRESS INVERT
 98   ITNINV=INVF/2
      CALL MSSG(40HSTARTED FROM GIVEN INVERSE ON INPUT          )
C-----RESET INPUT FILE FOR MAPOUT TO OVERWRITE LAST INVERSE
      BACKSPACE INPUT
      BACKSPACE INPUT
      IF(L.EQ.0) GOTO 10
      BACKSPACE INPUT
      GOTO 10
C
```

```
C
      ENTRY INMAP
C-----READS MAP CARDS FROM INPUT TO FILE IMAP AND TERMINATES THEM
      CALL MSSG(40HINMAP LOOKED FOR  MAP                           )
      DO 200 I=1,1000
      READ(5,2) CARD
      IF(ENDFILE 5) 201,199
 199  CONTINUE
 2    FORMAT(8A10)
      IF (CARD(1).EQ.ENDER) GOTO 201
      IF(CARD(1).EQ.REWTND)  GOTO 202
      WRITE(IMAP,2) CARD
 1999 CONTINUE
 200  CONTINUE
C-----ENDS THE MAPOUT CARDS
 201  WRITE(IMAP,2)  ENDER
      RETURN
 202  REWIND IMAP
      CALL MESSG(40HINMAP--DELETED EXISTING MAP,  IF ANY          )
      GOTO 1999
      END
      SUBROUTINE MAPOUT(B)
C-----SUB VERSION  APRIL-20-71
C-----OUTPUTS THE FINAL BASIS FOR MAPIN USE
C
      COMMON /NAMES/ NAME(100)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON/IXX/IX(100) /XX/ X(100)
      COMMON /INPUT/INPUT,INPUTM,INPUTN
      COMMON /FILES/ IA1,IA2,IMAP
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS
      EQUIVALENCE (MAPKEY,AJ)
      DIMENSION MAPBAS(100),MAP BND(10),MAPNLL(10),MAPKEY(1000)
      REAL B(1)
C
      CALL MESSG(40HMAPOUT                                  )
      REWIND IMAP
      JNCORE=0
      DO 50 I=1,MC
      IF(NAME(I).NE.2) GOTO 50
      WRITE(IMAP,1) I
    1 FORMAT(10HBASICROWS  ,I10)
   50 CONTINUE
C
      K=INLL=IBND=IBAS=IKEY=0
C-----CLEAR SOLUTION SPACE
      NVARS=INPUTM+NBDS
      DO 60 I=1,NVARS
      IX(I)=0
   60 X(I)=0.
      MP1=MC+1
      DO 40 I=MP1,NT
      JCOL=I-MC
      J=MOD(NAME(I),10)+1
      GOTO(10,40,20,30,35),J
C
   10 INLL=INLL+1
      MAPNLL(INLL)=JCOL
      GOTO 40
   20 IBAS=IBAS+1
      MAPBAS(IBAS)=JCOL
      DO 25 IR=1,M
      IF(IBASIS(IR)/100.EQ.I) GOTO 26
   25 CONTINUE
   26 K=K+1
      IX(K)=JCOL
      X(K)=BETA(IR)
      GOTO 40
   30 IBND=IBND+1
      MAPBND(IBND)=JCOL
      K=K+1
      IX(K)=JCOL
      X(K)=BOUND(I)
      GOTO 40
   35 IKEY=IKEY+1
      MAPKEY(IKEY)=JCOL
      DO 36 IR=1,L
      IF(KEYS(IR)/100.EQ.I) GOTO 37
   36 CONTINUE
   37 K=K+1
      IX(K)=JCOL
      X(K)=BETA(IR+M)
   40 CONTINUE
      IF(IBAS.NE.0) WRITE(IMAP,2) (MAPBAS(I),I=1,IBAS)
      IF(IBND.NE.0) WRITE(IMAP,3) (MAPBND(I),I=1,IBND)
      IF(INLL.NE.0) WRITE(IMAP,4) (MAPNLL(I),I=1,INLL)
      IF(IKEY.NE.0) WRITE(IMAP,6) (MAPKEY(I),I=1,IKEY)
    2 FORMAT(10HBASIC     ,4I10)
    3 FORMAT(10HATBND     ,4I10)
    4 FORMAT(10HNULL      ,4I10)
    5 FORMAT(10HEND       )
    6 FORMAT(10HKEY       ,4I10)
    7 FORMAT(10HINVERSE   )
      IF(MOD(K3,2).NE.0) GOTO 598
C-----PLACE BASIS ON END OF INPUT TAPE, AFTER ANY THERE ALREADY
      MM=M*M
      WRITE(INPUT) (B(T),I=1,MM)
      WRITE(INPUT) (IBASIS(J),BETA(J),J=1,M)
      IF(L.NE.0) WRITE(INPUT) (KEYS(J),BETA(J+M),J=1,L)
      WRITE(IMAP,7)
  598 WRITE(IMAP,5)
  599 IF(MOD(K3,5).NE.0) RETURN
  600 WRITE(6,601)
  601 FORMAT(*0CURRENT SOLUTION*/*0       BASIS        VALUE         -PI*)
      WRITE(6,602) (IBASIS(I),BETA(I),B(IPI+I-1),I=1,M)
  602 FORMAT(I12,2E12.4)
      WRITE(6,603) (KEYS(I),BETA(I+M),I=1,L)
  603 FORMAT(*0         KEYS        VALUE*/(I12,E12.4))
      WRITE(6,604) (IX(I),X(I),I=1,K)
  604 FORMAT(*0SOLUTION VECTOR, PACKED*/(I12,E12.4))
C-----PRICE OUT REMAINING VECORS
      WRITE(6,701)
  701 FORMAT(*0REMAINING VECTORS*)
      DO 700 J=MP1,NT
      JTYPE=MOD(NAME(J),10)
      IF(JTYPE.EQ.2) GOTO 700
      CALL IN(J,GAMMA,1)
      DJVAL=DOTS(M,B(IPI),GAMMA)
      WRITE(6,702) J,DJVAL,NAME(J)
  702 FORMAT(I12,12X,E12.4,I12 )
  700 CONTINUE
      RETURN
      END
      SUBROUTINE PIVOT(IROW,B,ALPHA)
C-----PIVOT ALPHA INTO B ROW IROW
      COMMON /NAMES/ NAME(100)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /STATE/ JPOS
      COMMON /T/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      REAL ALPHA(1),B(1)
      NPKT(I)=MOD(NAME(I),100000)/10
C
C-----CHECK ALPHA HAS A ROW
      IF(IROW.EQ.0) GOTO 90
C-----NORMALISE ROW  IROW
 1    CONTINUE
      IORG=M*(IROW-1)
      PIV=1.
      IF (ALPHA(IROW).EQ.1.0)  GOTO   20
      IF( ABS(ALPHA(IROW) ).GT.PIVTOL) GOTO 5
      CALL ERROR(40HPIVOT--PIVOT LESS THAN PIVTOL                   )
      CALL ESCAPE(B)
 5    PIV   = 1.0/ALPHA(IROW)
      DO 10 I=1,M
 10   B(IORG+I)= B(IORG+I)*PIV
C-----PIVOT OP FOR ROW I,  LEAVE  ALPHA.
 20   DO 30 I=1,M
      IF( I.EQ.IROW) GOTO 30
      IF( ABS( ALPHA(I)).LT.ZERO) GOTO 30
      JORG=M*(I-1)
      PIV=ALPHA(I)
      DO 25 J=1,M
 25   B(JORG+J)=B(JORG+J)-PIV *B(IORG+J)
 30   CONTINUE
      RETURN
C
C-----FIND BEST ROW TO PIVOT ALPHA INTO B
 90   CONTINUE
      PIV=PIVTOL
      DO 100 I=1,M
C-----CHECK FOR FREE LOGICALS
      JP=IBASIS(I)/100
      IF(JP.NE.JPOS) GOTO 99
      IROW=I
      GOTO 1
 99   IF(JP.NE.0) GOTO 100
C-----ZERO BASIS ENTRY AT I
      DIVOT=ABS(ALPHA(I))
      IF(DIVOT.LT.PIV) GOTO 100
      PIV=DIVOT
      IROW=I
 100  CONTINUE
      IF(IROW.EQ.0) GOTO 150
C-----BEST ROW TO ADD THIS COLUMN IS IROW
 101  CONTINUE
      IBASIS(IROW)=100*JPOS+IBASIS(IROW)
      GOTO 1
C-----THE COLUMN IS NO GOOD ANYWHERE AT PRESENT, DROP FROM BASIC SET
```

```
  150 CONTINUE                                                          20020290
      CALL SETBNB(-JPOS)                                                20020300
      IF(MOD(K3,13).NE.0) RETURN                                        20020310
      WRITE(6,151)  JPOS                                                20020320
  151 FORMAT(1H ,*PIVOT DROPPED COLUMN* I6     )                        20020330
      RETURN                                                            20020340
      END                                                               20020350
      SUBROUTINE PRIMAL(B)
C------GUB VERSION APRIL/71
      COMMON  /MOVES/ THETA,BNDJ,DMAX,PRMLER,DUALER
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /A/ ALPHA(101) /B/ BETA(1 1) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /DJS/ DJ(100)
      COMMON /NAMES/ NAME(100)
      COMMON /RHS/ RHS(100)
      REAL B(1)
      LOGICAL BASIC,ATBND
      ITP(I)=MOD(IBASIS(I),10)
      ATBND(I)=MOD(NAME(I),10).EQ.3
      NPKT(J)=MOD(NAME(J),100000)/10
C
      CALL MESSG(40HPRIMAL                                  )
C
      IROW=JCOL=NREJ=NDEG=0
      CALL STATUS(40HPRIMAL--BEGIN                          )
 3000 CONTINUE
C-----FIND THE COST ROW
      IC=ICOST
      IF ( IPHASE.EQ.1 )  IC= M
C-----KEEP PHASE1 COST ZERO IN PHASE 2
C-----PHASE 1 COST IS FREE IN PHASE 1
      IBASIS(M)=100*(IBASIS(M)/100)
      IF(IPHASE.EQ.1) IBASIS(M)=IBASIS(M)+3
C-----PICK UP NEW PI
      IPI=1+M*IC-M
C-----CUTOFF FOR DEGENERACY REJECTS
      NDEGLM=0
C
C*****BASIC CYCLE OF 2 PHASE LP***********
 1    ITRN=1+ITRN
      THETA=BNDJ=IROW=JCOL=JOUT=JPOS=0
C
C-----LOCATE PIVOTAL COLUMN
 30   CALL COLUMN( JCOL,B)
      IF( JCOL.NE.0)  GOTO 50
      CALL XCHECK(6HPRIMAL,4HQUIT,B)
C
C-----OPTIMUM, CHECK MODE  = PHASE1/ PHASE2/ NOFEAS
      IF( IPHASE.EQ.2)  GOTO 2000
      IF(  ABS(BETA(IC)).LT.CTOL) GOTO 1000
      CALL  NO FEAS
      RETURN
C
C-----STEP PROCEDURE FOR  IN CORE COLUMN  JCOL
 50   CONTINUE
C-----LOCATE COLUMN  POSITION AND BOUND
      JPOS = JA(JCOL)
      BNDJ = BOUND(JPOS)
```

```
C
C-----LOCATE PIVOTAL ROW IROW AND STEP THETA
      CALL  ROW(THETA, IROW, JCOL, ITYPE,B)
      IF( IROW.NE.0)  GOTO 60
      CALL XCHECK(6HPRIMAL,4HQUIT,B)
      CALL UNBND
      RETURN
C
C
C-----DEGENERACY AND PIVOT CHECKS
 60   IF( ABS(ALPHA(IROW)).GE.PIVTOL)  GOTO 65
      IF(NREJ.LT.2) GOTO 62
      WRITE(6,61) JA(JCOL),IROW, ALPHA(IROW),BETA(IROW)
 61   FORMAT(20X,*REJECTED COLUMN*I5*  ROW*I5*  PIVOT*E12.4*  RHS*E12.4)
 62   CONTINUE
      JAREJ(JCOL)=1
      NREJ = 1+NREJ
      IF(NREJ.NE.5) GOTO 64
C-----RE-INVERT, CLEAR REJECTS AND CONTINUE
      ITNINV=ITRN
      CALL INVERT(B)
      DO 63 J=1,JNCORE
 63   JAREJ(J)=0
 64   CONTINUE
      IF(NREJ.LT.100) GOTO 30
      CALL ERROR(40HPRIMAL--TOO MANY REJECT VECTORS                 )
C-----TRY ENDING IF PHASE 2
      IF(IPHASE.EQ.2) GOTO 2000
      CALL ESCAPE(B)
C
 65   IF( ABS(THETA*DJ(JCOL)).GE.CTOL) GOTO 70
      IF(NDJS.EQ.1) GOTO 70
      IF(NDEG.GE.NDEGLM) GOTO 70
      NDEG=1+NDEG
      JAREJ(JCOL)=1
      GOTO 30
C
C-----CHECK EXCEED BOUND ON JPOS---- XJPOS MOVES TO OR OFF  BOUND
 70   CONTINUE
      CALL XCHECK(6HPRIMAL,3HEND,B)
      IF( ABS(THETA)+ZERO.LT.BNDJ)  GOTO 80
      ITYPE=1
C-----SUPPRESS PRICING NEXT TIME
      NREJ=1
C-----JOUT=0 KILLS STATUS PRINT
      JOUT=0
      IF(THETA.GE.0.0) GOTO 75
C-----XJPOS COMES OFF BOUND , THETA NEG.
      CALL SETBND(  -JPOS  )
      THETA= - BNDJ
      GOTO 90
C-----XJPOS GOES TO BOUND
 75   CALL SETBND( JPOS)
      THETA = BNDJ
      GOTO  90
C
C-----PICK UP REJECTED COL, CHECK FOR KEY CHANGE
 80   CONTINUE
```

```
      JOUT=IBASIS(IROW)/100
      IF(IROW.LE.M) GOTO  800
C-----KEY CHANGE, CORRECT REJECTED COL AND CHECK ESSENTIL PACKET
      JOUT=KEYS(IROW-M)/100
      NBVPKT=MOD( KEYS(IROW-M),100)
      IF(NBVPKT.GT.0)GOTO  81
C-----CHANGE KEY FROM JOUT TO JPOS IN NON-ESSENTIAL PKT
      KEYS(IROW-M) = 100*JPOS
      CALL SET KEY(JPOS)
      CALL SET KEY(-JOUT)
C-----SET PARAMS FOR KEY STEP
      EPSI=THETA
C-----SUPPRESS PRICING NEXT TIME
      NREJ=1
      GOTO  90
C
C-----ESSENTIAL PACKET, CHANGE JOUT FROM KEY TO BASIC IN NEWROW
 81   CALL KEY CH(JOUT,NEWROW,B)
      IROW = NEWROW
C
C-----NORMAL PIVOT OPERATION
 800  CALL PIVOT(IROW,B, ALPHA )
C-----UPDATE KEY BASIS COUNTS FOR JPOS AND JOUT
      JPOSPK=NPKT(JPOS)
      IF(JPOSPK.EQ.0) GOTO  82
      KEYS(JPOSPK)= KEYS(JPOSPK)+1
 82   JOUTPK=NPKT(JOUT)
      IF(JOUT.GT.NT) GOTO 84
      IF(JOUTPK.EQ.0) GOTO  84
      KEYS(JOUTPK)= KEYS(JOUTPK)-1
 84   CONTINUE
C
C-----CHECK JPOS COMING OFF A BOUND (THETA.LE.0 )
      EPSI=THETA
      IF(ATBND(JPOS))  EPSI=BNDJ+THETA
C
C-----CHECK JOUT, MARK NEW AND UNMARK OLD BASIC COLS
      JOUT = IBASIS( IROW )/100
      IBASIS(IROW) = 100*JPOS+MOD(IBASIS(IROW),100)
      CALL SETBNB( JPOS)
      CALL SETBNB(-JOUT)
C-----ITYPE=2  IMPLIES JOUT OFF BOUND,  =3  IMPLIES  JOUT  TO BOUND
      IF(ITYPE.EQ.3) CALL SETBND(JOUT)
C-----RELEASE REJECTED VECTORS AFTER A PIVOT
      IF(NREJ+NDEG.EQ.0) GOTO 90
      NREJ=NDEG=0
      DO 85 I=1,JNCORE
C
C
C-----STEP BETA AND CONDITION COMPLETE PROBLEM  (GUB ROWS ARE LAST)
 85   JAREJ(I)=0
 90   DO  100 I=1,MPL
      BETA(I)=BETA(I)-THETA*ALPHA(I)
 100  CONTINUE
      IF(ITYPE.NE.1) BETA(IROW)= EPST
      DO 110 I=1,M
      IF(ITP(I).EQ.3) GOTO 110
      IF(BETA(I).GE.0.0) GOTO 110
      BETA(I)=0.0
  110 CONTINUE
      DO 120 I=1,L
      IF(BETA(I+M).GE.0.0) GOTO 120
      BETA(I+M)=0.0
  120 CONTINUE
      CALL STATUS(40HEND OF PRIMAL                           )
      GOTO 1
C*****END OF BASIC PRIMAL CYCLE****************
C
C-----OPTIMUM PHASE1 TERMINATION
 1000 CONTINUE
      CALL OPT1
      IPHASE=2
      BETA(M)=0.0
      GOTO 3000
C
C-----OPTIMUM PHASE2 TERMINATION
 2000 CONTINUE
      CALL OPT2
C
      RETURN
      END
      SUBROUTINE  ROW(THETA,IROW,JCOL,ITYPE,B)
C-----GUR VERSION  APRIL 20/71
C-----FINDS STEP TO BOUND AND BOUND ENCOUNTERED
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPI
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /NAMES/ NAME(100)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      LOGICAL BASIC,ATBND
      REAL B(1)
      ITP(J)=MOD(IBASIS(J),100)
      NPKT(J)=MOD(NAME(J),100000)/10
      ATBND(I)=MOD(NAME(I),10).EQ.3
C
C-----TRANSFORM SELECTED COLUMN
      JPOS= JA(JCOL)
C-----LOAD  COLUMN  TO DELTA (MAYBE NULL PACKET)
      JORG= M*JCOL-M
      DO 5  I=1,M
 5    DELTA(I)= AJ(JORG+I)
C-----NULL ELEMENTS IN LOWER ALPHA FOR PACKET ROWS
      DO 6  I=1,L
 6    ALPHA(I+M)=0.
C-----PACKET ROW HAS A UNITY
      JPKT= NPKT(JPOS)
      IF (JPKT.EQ.0)  GOTO 16
      ALPHA(JPKT+M)=1.
C-----FIND KEY AND KORG
      KORG=KEYFND(JPKT)
      IF(KORG.NE.0) GOTO 14
      CALL ERROR(40HROW--KEY NOT IN CORE                          )
      WRITE(6,998)JPKT,JCOL,JPOS
 998  FORMAT(1H+,40X,*PACKET*I5*  POSITION*I5*  COLUMN*I5)
      CALL ESCAPE(B)
      IROW=0
      RETURN
 14   CONTINUE
      KORG=M*KORG-M
      DO 15 I=1,M
 15   DELTA(I)=DELTA(I)-AJ(KORG+I)
C-----TRANSFORM REDUCED  COLUMN  IN LOWER ALPHA
 16   IORG=1
      DO 20 I=1,M
      ALPHA(I)  = DOT( M, B(IORG), DELTA)
 20   IORG=IORG+M
C-----SUM PACKET BASIC ENTRIES TO  ALPHA ELEMENTS
      IF(L.EQ.0) GOTO 26
      DO  25 I=1,M
      IB = IBASIS(I)/100
      IF(IB.GT.NT) GOTO 25
      K= NPKT( IB )
      IF(K.EQ.0) GOTO 25
      K=K+M
      ALPHA(K)=ALPHA(K)-ALPHA(I)
 25   CONTINUE
 26   CONTINUE
```

```
C
      THETA=1.E35
      IROW = 0
      ITYPE= 1
C
      IF(  ATBND(JPOS) )  GOTO 100
C
C-----X(JPOS) ZERO,  DJ  NEGATIVE ---INCREASE   X(JPOS)
      DO  50 I=1,MPL
      IF(I.LE.M .AND. ITP(I).EQ.3) GOTO 50
      IF (  ALPHA(I).LT.-ZERO)   GOTO  30
      IF (  ALPHA(I).GT. ZERO)   GOTO  10
      GOTO 50
C-----POSITIVE PIVOT
 10   STEP  = BETA(I)/ALPHA(I)
      IF(  STEP .GE. THETA )  GOTO   50
      THETA = STEP
      IROW  = I
      ITYPE = 2
      GOTO  50
C-----NEGATIVE PIVOT-----(  BOUND(JOUT).GE.BETA(I) )
 30   JOUT  =  IBASIS(I)/100
      IF(I.GT.M) JOUT=KEYS(I-M)/100
      STEP  =  ( BETA(I) - BOUND(JOUT) ) /  ALPHA(I)
      IF(  STEP .GE. THETA )  GOTO   50
      THETA =  STEP
      IROW  =  I
      ITYPE =  3
 50   CONTINUE
      GOTO  200
C
C-----X(JPOS) AT BOUND   DJ POS.--DECREASE  X(JPOS)
 100  DO 150  I=1,MPL
      IF(I.LE.M .AND. ITP(I).EQ.3) GOTO 150
      IF(  ALPHA(I).LT. - ZERO )    GOTO 130
      IF(  ALPHA(I).GT.   ZERO )    GOTO 110
      GOTO  150
C-----POSITIVE PIVOT----(  BOUND(JOUT).GE. BETA(I) )
 110  JOUT =  IBASIS(I)/100
      IF(I.GT.M) JOUT=KEYS(I-M)/100
      STEP= (  BETA(I)-BOUND(JOUT) )/(-ALPHA(I))
      IF( STEP. GE . THETA  )   GOTO  150
      THETA = STEP
      IROW  = I
      ITYPE = 3
      GOTO    150
C-----NEGATIVE PIVOT
 130  STEP=  BETA(I)/( -ALPHA(I) )
      IF( STEP  .GE. THETA)     GOTO  150
      THETA=  STEP
      IROW =  I
      ITYPE=  2
 150  CONTINUE
      THETA= -THETA
C-----PIVOTS ON 2,3,  2 DRIVES JOUT TO  ZERO , 3 MOVES JOUT TO BOUND
C
 200  RETURN
      END
      SUBROUTINE SETBND(I)
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /NAMES/ NAME(100)
      K=3
  100 CONTINUE
      IF(I) 1,2,3
    1 J=-I
      IF(J.LE.NT) NAME(J)=10*(NAME(J)/10)+1
    2 RETURN
    3 IF(I.LE.NT) NAME(I)=10*(NAME(I)/10)+K
      RETURN
C
      ENTRY SETBNB
      K=2
      GOTO 100
C
      ENTRY SETNNN
      K=0
      GOTO 100
C
      ENTRY SETKEY
      K=4
      GOTO 100
      END
      SUBROUTINE SETUP                                                  20023710
C------GUB VERSION APRIL/71                                             20023720
      INTEGER  PKT,PKT1                                                 20023730
      COMMON /INPUT/INPUT,INPUTM,INPUTN                                 20023740
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN                     20023750
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI                 20023760
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101) 20023770
      COMMON /ROWTYP/ IROWTP(101) /NAMES/ NAME(100)                     20023780
      COMMON /BASIS/ IBASIS(101),KEYS(101)                              20023790
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)                20023800
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,NWAJ                 20023810
      COMMON /RHS/ RHS(100)                                             20023820
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL                20023830
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS            20023840
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS                        20023850
      COMMON / CV2 / T(100,10),BO(100),BLO(10),UBS(10),CO(10)           20023860
      COMMON /CV3/ MBBCAV,NBBCAV,NCHGS                                  20023870
C-----TAKES  INPUT A MATRIX IN COLUMNS OFF INPUT FILE BY COLUMNS         20023880
C-----INITIAL SETUP FOR COMPLETE PROBLEM IS CHANGED AT END TO REDUCED PR20023890
C-----OUT DROPS GUB ROWS AS FOUND                                       20023900
      CALL MESSG(40HSETUP                                   )           20023910
C-----ACTUAL PROBLEM SIZES                                              20023920
      M= INPUTM+1                                                       20023930
C-----TRY TO START IN PHASE 2                                           20023940
      IPHASE=2                                                          20023950
      IC=ICOST                                                          20023960
C-----TOLERANCES                                                        20023970
      DJTOL=1.E-8                                                       20023980
      ZERO=1.E-12                                                       20023990
      PIVTOL=1.E-5                                                      20024000
      CTOL=1.E-4                                                        20024010
      PERTOL=1.E-5                                                      20024020
      DERTOL=1.E-8                                                      20024030
      NSCAN=NTRY=MAXTRY=NDJS=ITRN=JNCORE=0                              20024040
C-----INVERT FREQUENCY INVF, ITERATION OF NEXT INVERT ITNINV            20024050
      INVF=100                                                          20024060
      ITNINV=0                                                          20024070
C                                                                       20024080
C                                                                       20024090
C-----FIRST M COLUMNS GIVE ROW LOGICALS AND TYPES                       20024100
      DO 5 I=1,M                                                        20024110
      KEYS(I)=0                                                         20024120
 5    IBASIS(I)=0                                                       20024130
C-----MARK PHASE1 COST ROW FREE FOR POSSIBLE USE IN PHASE 1             20024140
      IROWTP(M)=3                                                       20024150
C-----SET UP LOGICAL VECTORS FOR THE ROWS                               20024160
      DO  10 I=1,M                                                      20024170
 10   ALPHA(I)=0.0                                                      20024180
C-----NOW COUNT COLS WRITTEN                                            20024190
      NT=0                                                              20024200
      DO 100 I=1,M                                                      20024210
      ID=IROWTP(I)                                                      20024220
      IF(ID.EQ.0)  GOTO 20                                              20024230
      IF(ID.EQ.1)  GOTO 21                                              20024240
      IF(ID.EQ.2)  GOTO 22                                              20024250
      IF(ID.EQ.3)  GOTO 23                                              20024260
      IF(ID.EQ.4) GOTO 24                                               20024270
 18    CALL ERROR(40H  SETUP--ROW TYPE ERROR--OUT OF RANGE                )
       CALL ESCAPE
C-----EQUALITY ROW- NO LOGICAL COL.
 20    GOTO 100
C-----LE. ROW- POSITIVE LOGICAL+SLACK
 21    ALPHA(I)=1.0
       K=1
       GOTO 50
C-----GE. ROW- NEGATIVE LOGICAL+SLACK
 22    ALPHA(I)=-1
       K=1
       GOTO 50
C-----FREE ROW-POSITIVE LOGICAL-BASIC
 23    ALPHA(I)=+1.
       IBASIS(I)=NT+1
       K=2
       GOTO 50
C-----SUB ROW-NO LOGICAL
 24    CONTINUE
       GOTO 100
C-----PLACE COLUMN IN FILE IA1 AND IA2
   50  NT=NT+1
       NAME(NT)=K
       CALL OUT(NT,ALPHA,COLNM)
       ALPHA(I)=0
 100   CONTINUE
C-----KEEP PHASE 1 COST ROW ZERO IN PHASE 2
       IROWTP(M)=0
C-----NO. OF LOGICAL COLS MC
       MC=NT
C
C-----CYCLE INPUT FILE COLUMS
       REWIND INPUT
       DO 200 JNT=1,INPUTN
 110   IF(JNT.NE.JRHS)   GOTO 130
       READ (INPUT)   (RHS(J),J=1,INPUTM)
C-----TAKE RHS FROM BO AND SKIP TAPE VERSION
       DO 120 J=1,INPUTM
 120   RHS(J)=BO(J)
       RHS(M)=0.
       NT=NT+1
       NAME(NT)=0
       CALL OUT(NT,RHS,COLNM)
       GOTO 200
C-----GET NEXT COLUMN JNT
130    CONTINUE
       READ(INPUT)(ALPHA(J),J=1,INPUTM)
C-----INSERT COLUMN CHANGES TO PROBLEM
       IF(JNT.GT.NCHGS)  GOTO 135
       ALPHA(ICOST)=CO(JNT)
 135   CONTINUE
C-----CHECK FOR COL PACKET, GET PKT NO. OR 0
       PKT=0
       PKT1=0
       DO 140 I=1,INPUTM
       IF( IROWTP(I).NE.4)   GOTO 140
       PKT1=1+PKT1
       IF( ALPHA (I).NE.1.) GOTO 140
```

```
      PKT =PKT1
      GOTO 145
 140  CONTINUE
C-----CHECK FOR BOUND, GET BOUND NO. OR 0
 145  IF(NBDS.EQ.0) GOTO 151
      DO 150 J=1,NBDS
      IF ( IBDS(J).EQ.JNT) GOTO 155
 150  CONTINUE
 151  J=0
 155  CONTINUE
C-----SET NAME TO BOUND+PACKET + STATE  AND MARK KEY COLUMN
      K=1
C-----COUNT COLUMN AND WRITE TO FILE LESS GUB ELEMENTS
 160  NT=1+NT
      NAME(NT)=K+10*PKT+100000*J
      CALL OUT(NT, ALPHA,COLNM)
 200  CONTINUE
C-----REMOVE GUB ROWS FROM  IBASIS AND RHS
      INON=0
      L=0
      IKOST=ICOST
      DO 220 I=1,M
      IF(IROWTP(I).EQ.4) GOTO 210
C-----NON-GUB ROE
      INON=INON+1
      IBASIS(INON)=100*IBASIS(I)+IROWTP(I)
      RHS(INON)=RHS(I)
      GOTO 220
C-----GUB ROW, STORE RHS IN AJ
 210  L=L+1
      AJ(L)=RHS(I)
C-----MOVE DOWN USER COST ROW
      IF(I.LT.ICOST) IKOST=IKOST-1
 220  CONTINUE
C-----NOW REPLACE RHS ON END OF RHS
      IF(L.EQ.0) GOTO 240
      DO 230 I=1,L
 230  RHS(INON+I)=AJ(I)
C-----NOW DROP COUNT OF GUB ROWS
      M=M-L
      ICOST=IKOST
C-----REDUCED PROBLEM NOW COMPLETE
 240  CONTINUE
      RETURN
      END
      SUBROUTINE STATUS(NOTE)
      DIMENSION NOTE(4)
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JPHS,IPI
      COMMON /NAMES/ NAME(100)
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /DJS/ DJ(100)
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      DATA JNTO/0/
      LOGICAL BASIC,ATRND
      IF(MOD(K3,11).NE.0) RETURN
      IF( MOD(ITRN  ,50).EQ.0) WRITE(6,1)
 1    FORMAT(1H1
     +           ,10H     PHASE
     +           ,10H      ITER
     +           ,10H       TRY
     +           ,15H  VAL OBJECTIVE
     +           ,10H      NDJS
     +           ,10H     NARTS
     +           ,15H    VALUE DJ IN
     +           ,10H    COL IN
     +           ,10H      CODE
     +           ,10H   COL OUT
     +           ,10H      CODE
     +           ,10H     NSCAN          )
C-----STORE CURRENT SOLUTION, QUIT OR CONTINUE
      CALL SECOND(X)
      IF(X.LT.TMAX.AND.ITRN.LT.K5) GOTO 999
      CALL MAPOUT(B)
      CALL EXIT
 999    CONTINUE
C-----COUNT ACTIVE ARTIFICIALS FOR STATUS DATA
      NPIF=0
      DO 15 I=1,M
 15   IF(IBASIS(I)/100.GT.NT) NPIF=1+NPIF
      COST=-BETA(IC)
      NCOLS=JNCORE
      IF(NTRY.NE.0) GOTO 20
      CALL INPOS(JNT)
      NCOLS=JNT-JNTO
      IF(NCOLS.LE.0) NCOLS=NCOLS+NT
 20   JNTO=JNT
      JNSCAN=10000*NSCAN+NCOLS
      MNTRY=1000*MAXTRY+NTRY
      IF(JCOL.EQ.0) GOTO 10
      IF(IROW.EQ.0) GOTO 10
      NJOUT=NAME(JOUT)
      IF(JOUT.EQ.0) NJOUT=0
      IF(JOUT.GT.NT) NJOUT=10000000*IROW
      WRITE(6,2) IPHASE,ITRN,MNTRY,COST,NDJS,NPIF,DJ(JCOL)
     +   ,JPOS,NAME(JPOS),JOUT,NJOUT,JNSCAN
 2    FORMAT(1H ,3I10,F15.6,2I10,E15.6,5I10)
      RETURN
C-----WHEN NO COLUMN WAS SELECTED
```

```
C-----OR UNBOUNDED
 10    WRITE(6,3) IPHASE,IIRN,MNTRY,COST,NDJS,NPIF,NOTE,JNSCAN
 3     FORMAT(1H ,3I10,E15.6,2I10,15H---------------,4A10,I10)
       RETURN
C
C
       ENTRY ERROR
       WRITE(6,4) NOTE,NOTE,NOTE
 4     FORMAT(1H /(1H+,4A10))
       RETURN
C
C
       ENTRY MESSG
       ENTRY MSSG
       IF(MOD(K3,7).NE.0) RETURN
       CALL SECOND(X)
       WRITE(6,5) NOTE,X
 5     FORMAT((1H ,4A10,60X,F10.0,*  SECONDS* )
       RETURN
       END
```

```
      SUBROUTINE XCHECK(CALLER,AT,B)
      REAL B(1)
C-----GIVES QUICK CROSS CHECKS AND LOCATION
      COMMON /RHS/ RHS(100)
      COMMON  /MOVES/ THETA,BNDJ,DMAX,PRMLER,DUALER
      COMMON /PARAMS/ TMAX,ITNINV,INVF,K1,K2,K3,K4,K5
      COMMON /A/ ALPHA(101) /B/ BETA(101) /C/ GAMMA(101) /D/ DELTA(101)
      COMMON /BASIS/ IBASIS(101),KEYS(101)
      COMMON /CORE/ JAREJ(101),JA(101),JAK(101),AJ(1000)
      COMMON /I/ M,L,MPL,MC,NT,ICOST,IC,IPHASE,JRHS,IPI
      COMMON /LIMS/ MAXTRY,NTRY,JNCORE,NCRMAX,NSCAN
      COMMON /DJS/ DJ(100)
      COMMON /NAMES/ NAME(100)
      COMMON /TOLS/ DJTOL,ZERO,PIVTOL,CTOL,PERTOL,DERTOL
      COMMON /STATE/ JPOS,IROW,JCOL,JOUT,ITRN,NREJ,NPIF,NDJS
      COMMON /BOUNDS/ BOUNDS(100),IBDS(100),NBDS
      INTEGER DELTA
      IROWTP(I)=MOD(IBASIS(I),100)
C-----K4 IS 1000*START + STOP ITERATION FOR XCHECKS
      IF(K4/1000 .GT.ITRN .OR. MOD(K4,1000).LT.ITRN) RETURN
      K4=1000*(ITRN+K2)+MOD(K4,1000)
      WRITE(6,1) CALLER,IROW,THETA,BNDJ,JPOS
    1 FORMAT(*0...................................*/
     +        * XCHECK CALLED BY *A6*  PIVOT ROW---*I3,
     +        *  STEP*E12.5*  BOUND*E11.4*  COLUMN*I5)
      WRITE(6,6) (DJ(J),J=1,JNCORE)
    6 FORMAT(32X,10F10.4)
      JKOL=JCOL
      J1=MAX0(JKOL-5,1)
      J2=MIN0(J1+9,JNCORE)
      WRITE(6,5)   (JA(K),K=J1,J2)
    5 FORMAT(32X,10I10)
      IORG=1
      JEND=JNCORE*M
      DO 40 I=1,M
      JORG=J1*M-M+1
      DO 30 J=J1,J2
      AJ(JEND+J)=DOT(M,B(IORG),AJ(JORG) )
      IF( ABS(AJ(JEND+J)).LE.ZERO ) AJ(JEND+J)=0.
   30 JORG=JORG+M
      WRITE(6,8) I,IBASIS(I),ALPHA(I),BETA(I),(AJ(JEND+J),J=J1,J2)
   40 IORG=IORG+M
      DO 45 J=J1,J2
      JAJ=JA(J)
   45 DELTA(J)=NAME(JAJ)
      WRITE(6,5)(DELTA(J),J=J1,J2)
      IF(L.EQ.0) GOTO 60
      DO 50 I=1,L
      WRITE(6,8) I,KEYS(I),ALPHA(I+M),BETA(I+M)
   50 CONTINUE
   60 CONTINUE
    8 FORMAT(I3,I7,2E10.2,2X,10E10.2)
C-----ERROR CHECKING OPTION IN XCHECK
   70 CONTINUE
      IF(MOD(K3,17).NE.0) GOTO 555
C------GAMMA SUMS LHS-RHS
      DO 90 I=1,MPL
   90 GAMMA(I)=-RHS(I)
C-----CYCLE ALL BOOK KEEPING TO GET COLUMNS IN ORDER
      LAST=1
      NVARS=MPL+NBDS
      DO 500 NVAR=1,NVARS
      NEXT=999999999
C-----BASIC COLUMNS
      DO 200 I=1,M
      J=IBASIS(I)/100
C-----MARK STRUCTURALS
      ITAG=2
      IF(J.LE.NT) GOTO 150
C-----MARK ARTIFICIALS
      ITAG=5
      IF(J.LE.NT+100) GOTO 150
C-----MARK NEGATIVE STRUCTURALS
      ITAG=6
      J=J-(NT+100)
      IF(J.LE.NT) GOTO 150
C-----MARK NEGATIVE ARTIFICIALS
      ITAG=7
  150 CONTINUE
      IF(J.LT.LAST) GOTO 200
      IF(J.GT.NEXT) GOTO 200
      NEXT=J
      JTYPE=2
      JTAG=ITAG
      X=BETA(I)
      IBAS=I
  200 CONTINUE
C-----BOUNDED COLUMNS
      IF(NBDS.EQ.0) GOTO 300
      NEXTJ=MIN0(NT,NEXT)
      DO 250 J=LAST,NEXTJ
      IF(MOD(NAME(J),10).EQ.3) GOTO 290
  250 CONTINUE
      GOTO 300
  290 NEXT=J
      JTYPE=3
      X=BOUND(J)
C-----KEY COLUMNS
  300 CONTINUE
      IF(L.EQ.0) GOTO 360
      DO 350 I=1,L
      J=KEYS(I)/100
      IF(J.LT.LAST) GOTO 350
      IF(J.GT.NEXT) GOTO 350
      NEXT=J
      JTYPE=4
      X=BETA(M+I)
  350 CONTINUE
  360 CONTINUE
C
C-----GET NEXT COLUMN TO CORE IF REAL (JTAG=2 OR 6)
      IF(NEXT.EQ.999999999) GOTO 510
      IF(NEXT.GT.NT) GOTO 400
      CALL IN(NEXT,AJ(JEND+1),JNCORE+1)
C-----ADD GUB ELEMENTS
```

```
      MP1=M+1
      IF(L.EQ.0) GOTO  385
      DO 380 I=MP1,MPL
 380  AJ(JEND+I)=0.
      IGUB=MOD(NAME(NEXT),100000)/10
      IF(IGUB.NE.0) AJ(JEND+M+IGUB)=1.
 385  IF(JTYPE.NE.2) GOTO 450
      IF(JTAG.NE.5) GOTO 450
C-----NEGATIVE STRUCTURALS
      DO 390 I=1,MPL
 390  AJ(JEND+I)=-AJ(JEND+I)
      AJ(JEND+M)=1.
      GOTO 450
C-----ARTIFICIAL VECTOR (JTAG=5 OR 7)
 400  DO 410 I=1,MPL
 410  AJ(JEND+I)=0.
      AJ(JEND+IBAS)=1.
      AJ(JEND+M)=1.
      IF(JTAG.EQ.7) AJ(JEND+IBAS)=-1.
C
C-----SUM X*AJ TO GAMMA-- THE ERROR
 450  DO 460 I=1,MPL
 460  GAMMA(I)=GAMMA(I)+X*AJ(JEND+I)
 500  LAST=NEXT+1
 501  FORMAT(4I4,E12.4,10F10.4/(28X,10F10.4))
 510  CONTINUE
C
C-----CHECK ERROR AGAINST TOLERANCE
      PRMLER=0.
      K=0
      DO 550 I=1,MPL
      ABSGAM=ABS(GAMMA(I))
      IF(ABSGAM.LE.PERTOL) GOTO 550
      K=K+1
      DELTA(K)=I
      GAMMA(K)=GAMMA(I)
      IF(ABSGAM.LE.PRMLER) GOTO 550
      PRMLER=ABSGAM
 550  CONTINUE
      IF(PRMLER.LE.PERTOL) WRITE(6,552)
      IF(PRMLER.LE.PERTOL) GOTO 555
      WRITE(6,551) PRMLER,PERTOL,(DELTA(I),GAMMA(I),I=1,K)
      IF(MOD(K3,19).NE.0) GOTO 555
      ITNINV=ITRN
 551  FORMAT(*0PRIMAL ERRORS EXCEED TOLERANCE---*/
     +        * ERROR--*E12.4*   TOLERANCE--*E12.4/(4(I10,E20.8)))
 552  FORMAT(* ERRORS WITHIN TOLERANCE*)
 555  WRITE(6,7)
 7    FORMAT(1H ,40H---------------------------------------- //  )
      RETURN
      END
```

```
       PROGRAM REPGEN(INPUT,OUTPUT,TAPEA,TAPE5=INPUT,
     1 TAPE6=OUTPUT,TAPE9=TAPEA)
      COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),
     *  VCOST(10,5),NAMEN(10),COSTS(30,3)
      COMMON /BASICS/ CHAP(5000,4),CODE(20),PER(10),IYR(10),LYR(10)
      COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),
     *  PURCH(10,20),STOR(10,20),SALV(10,20),PROC(20),PROT(20)
      COMMON /PARAMS/ RDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST
      DATA CODE / 2H01,2H02,2H03,2H04,2H05,2H06,2H07,2H08,2H09,2H10,
     *              2H11,2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20/
      CALL SETUP
   50 READ(9,100)(TITLE(I),I=1,4)
  100 FORMAT(4A10)
      IF(EOF,9)7777,200
  200 RDTOT=0.0
      DO 300 I=1,20
      SALE(I)=0.0
      SAVE(I)=0.0
      PROC(I)=PROT(I)
      OANDM(I)=0.0
      DO 250 N=1,10
      PURCH(N,I)=0.0
      EXIST(N,I)=0.0
      STOR(N,I)=0.0
      SALV(N,I)=0.0
  250 CONTINUE
  300 CONTINUE
      CALL INSOLN
      CALL CINFO
      CALL PINFO
      GO TO 50
 7777 STOP
      END
      SUBROUTINE CINFO
      COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),
     *  VCOST(10,5),NAMEN(10),COSTS(30,3)
      COMMON /BASICS/ CHAR(5000,4),CODE(20),PER(10),IYR(10),LYR(10)
      COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),
     *  PURCH(10,20),STOP(10,20),SALV(10,20),PROC(20),PROT(20)
      COMMON /PARAMS/ RDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST
      DATA  ONE,TOTAL,PERIOD / 2H01,6HTOTAL ,6HPERIOD /
      WRITE(6,100)(TITLE(I),I=1,4)
      SUMT=0.0
      TPROC=0.0
      TCOST=0.0
      DO 500 I=1,20
  500 TCOST=TCOST+OANDM(I)+SAVE(I)-SALE(I)
      TCOST=COST-TCOST-RDTOT
      DO 600 I=1,NP
  600 TPROC=TPROC+PROC(I)
      TPROC=TPROC/TCOST
      DO 1000 I=1,NP
      IF(PER(I).EQ.ONE)  N=I
 1000 CONTINUE
      DO 2000 I=N,NP
      K1=IYR(I)-INYR+1
      K2=LYR(I)-INYR+1
      DO 1500 K=K1,K2
      IF(K.EQ.K1) GO TO 1500
      OANDM(K1)=OANDM(K1)+OANDM(K)
      SALE(K1)=SALE(K1)+SALE(K)
 1500 CONTINUE
      J=I-N+1
      PROC(I)=PROC(I)/TPROC
      OANDM(K1)=OANDM(K1)+SAVE(I)
      SUM=PROC(I)+OANDM(K1)-SALE(K1)
      WRITE(6,200) PERIOD,CODE(J),PROC(I),OANDM(K1),SALE(K1),SUM
      IF (J.EQ.1) GO TO 2000
      OANDM(1)=OANDM(1)+OANDM(K1)
      SALE(1)=SALE(1)+SALE(K1)
      PROC(N)=PROC(N)+PROC(I)
 2000 SUMT=SUMT+SUM
      I=LYR(NP)-INYR+2
      WRITE(6,300) TOTAL,RDTOT,PROC(N),OANDM(1),SALE(1),SUMT
      WRITE(6,400) SALE(I)
  100 FORMAT(1H1,15X,4A10 / 1H-,*COST INFORMATION* /
     *1H-,12X,5(1H*,12X) / 13X,1H*,*   R AND D   *, 1H*,
     ** PROCUREMENT*,1H*,*  OPERATING *,1H*,*   SALVAGE   *,1H*,
     **     TOTAL    * / 1H ,77(1H*) / 13X,5(1H*,12X))
  200 FORMAT(1H ,A6,2X,A2,2X,1H*,12X,4(1H*,1X,F9.3,2X) / 13X,5(1H*,12X))
  300 FORMAT(13X,5(1H*,12X) / 1H ,A6,6X,5(1H*,1X,F9.3,2X) / 1H ,77(1H*))
  400 FORMAT(1H- / 1H-,*TRUNCATION VALUE FOR RESOURCES = *,F9.3)
      RETURN
      END
      SUBROUTINE INSOLN
      COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),
     *  VCOST(10,5),NAMFN(10),COSTS(30,3)
      COMMON /BASICS/ CHAR(5000,4),CODE(20),PER(10),IYR(10),LYR(10)
      COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),
     *  PURCH(10,20),STOR(10,20),SALV(10,20),PROC(20),PROT(20)
      COMMON /PARAMS/ RDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST
      INTEGER TOT
      TOT=NP+1
      DATA X/1HX/,W/1HW/,S/1HS/,BLANK/1H /,BLAN2/2H  /
 7777 READ(9,100)IND,VAL
      IF(IND.GE.0) GO TO 101
      COST=VAL
      GO TO 700
  101 IF(IND.EQ.0) GO TO 7777
      IF(CHAR(IND,1).EQ.X) GO TO 500
      IF(CHAR(IND,1).EQ.W) GO TO 400
      IF(CHAR(IND,1).EQ.S) GO TO 300
      IF(CHAR(IND,3).NE.BLAN2) GO TO 7777
C  INTERPRET PNN VARIABLES FOR INVESTMENT CONSTRAINTS
      DO 200 I=1,20
      IF (PER(I).EQ.CHAR(IND,2)) GO TO 210
  200 CONTINUE
      GO TO 1000
  210 PROC(I)=PROC(I)-VAL
      PROC(I+1)=PROC(I+1)+VAL
      GO TO 7777
C  INTERPRET INHERITED FLEET AND PURCHASE FLEET VARIABLES
  400 DO 405 J=1,20
      IF(CODE(J).EQ.CHAR(IND,2)) GO TO 410
  405 CONTINUE
      GO TO 1000
  410 DO 420 I=1,10
      IF(PER(I).EQ.CHAR(IND,3)) GO TO 430
  420 CONTINUE
      GO TO 1000
  430 ISTART=IYR(I)
      IF(CHAR(IND,1).EQ.X) PURCH(J,I)=VAL
      DO 450 I=1,10
      IF(PER(I).EQ.CHAR(IND,4)) GO TO 460
  450 CONTINUE
      GO TO 1000
  460 IEND=LYR(I)
      CALL VALUES(J,ISTART,IEND,VAL)
      GO TO 7777
C  INTERPRET MOTHBALL VARIABLES
  300 DO 350 I=1,20
      IF(CODE(I).EQ.CHAR(IND,2)) GO TO 360
  350 CONTINUE
      GO TO 1000
  360 DO 370 J=1,10
      IF(PER(J).EQ.CHAR(IND,3)) GO TO 380
  370 CONTINUE
      GO TO 1000
  380 STOR(I,J)=VAL
      LENP=LYR(J)-IYR(J)+1
      CALL MOTH(I)
      SAVE(J)=SAVE(J)+C*VAL
      GO TO 7777
C  INTERPRET MASTER VARIABLES
  500 IF(CHAR(IND,3).NE.BLAN2) GO TO 400
      DO 550 I=1,20
      IF(CODE(I).EQ.CHAR(IND,2)) GO TO 560
  550 CONTINUE
      GO TO 1000
  560 PURCH(I,TOT)=VAL
      IF(VAL.GT.0.0)
     *RDTOT=RDTOT+VCOST(I,3)
      GO TO 7777
C  ERROR MESSAGE
 1000 WRITE(6,600) (CHAR(IND,I),I=1,4)
      STOP
  700 RETURN
  100 FORMAT(I4,4X,F12.4)
  600 FORMAT(1H-,*ERROR IN INTERPRETATION OF *,A1,3A2)
      END
      SUBROUTINE PINFO
      COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),
     *  VCOST(10,5),NAMEN(10),COSTS(30,3)
      COMMON /BASICS/ CHAR(5000,4),CODE(20),PER(10),IYR(10),LYR(10)
      COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),
     *  PURCH(10,20),STOR(10,20),SALV(10,20),PROC(20),PROT(20)
      COMMON /PARAMS/ RDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST
      INTEGER TOT
      DATA TOTAL,PERIOD,BLANK / 6HTOTAL ,6HPERIOD,2H   /
      WRITE(6,1000)(TITLE(I),I=1,4)
      M=1
    5 GO TO (10,20,30,40,50,60,70,80,90),NV
   10 WRITE(6,1010)(VNAME(I),I=1,NV)
      GO TO 100
   20 WRITE(6,1020)(VNAME(I),I=1,NV)
      GO TO 100
   30 WRITE(6,1030)(VNAME(I),I=1,NV)
      GO TO 100
   40 WRITE(6,1040)(VNAME(I),I=1,NV)
      GO TO 100
   50 WRITE(6,1050)(VNAME(I),I=1,NV)
      GO TO 100
   60 WRITE(6,1060)(VNAME(I),I=1,NV)
      GO TO 100
   70 WRITE(6,1070)(VNAME(I),I=1,NV)
      GO TO 100
   80 WRITE(6,1080)(VNAME(I),I=1,NV)
      GO TO 100
   90 WRITE(6,1090)(VNAME(I),I=1,NV)
 1000 FORMAT(1H1,15X,4A10 / 1H-,*PURCHASED RESOURCES*)
 1010 FORMAT(1H-,12X,  1H*,12X  /13X,  1H*,2X,A8,2X /1H ,25(1H*)/
     * 13X,  1H*,12X)
 1020 FORMAT(1H-,12X,2(1H*,12X) /13X,2(1H*,2X,A8,2X)/1H ,38(1H*)/
     * 13X,2(1H*,12X) )
 1030 FORMAT(1H-,12X,3(1H*,12X) /13X,3(1H*,2X,A8,2X)/14 ,51(1H*)/
     * 13X,3(1H*,12X) )
 1040 FORMAT(1H-,12X,4(1H*,12X) /13X,4(1H*,2X,A8,2X)/1H ,64(1H*)/
     * 13X,4(1H*,12X) )
 1050 FORMAT(1H-,12X,5(1H*,12X) /13X,5(1H*,2X,A8,2X)/1H ,77(1H*)/
     * 13X,5(1H*,12X) )
 1060 FORMAT(1H-,12X,6(1H*,12X) /13X,6(1H*,2X,A8,2X)/1H ,90(1H*)/
     * 13X,6(1H*,12X) )
 1070 FORMAT(1H-,12X,7(1H*,12X) /13X,7(1H*,2X,A8,2X)/1H ,103(1H*)/
     * 13X,7(1H*,12X) )
 1080 FORMAT(1H-,12X,8(1H*,12X) /13X,8(1H*,2X,A8,2X)/1H ,116(1H*)/
     * 13X,8(1H*,12X) )
 1090 FORMAT(1H-,12X,9(1H*,12X) /13X,9(1H*,2X,A8,2X)/1H ,129(1H*)/
     * 13X,9(1H*,12X) )
  100 IF(M.GE.2) GO TO 305
      K=0
      DO 200 I=1,TOT
      IF(PER(I).EQ.CODE(1)) K=1
      IF(K.NE.1) GO TO 200
      TEMP1=PERIOD
      TEMP2=PER(I)
      IF(I.NE.TOT) GO TO 105
      TEMP1=TOTAL
      TEMP2=BLANK
  105 GO TO (110,120,130,140,150,160,170,180,190),NV
  110 WRITE(6,1110) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  120 WRITE(6,1120) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  130 WRITE(6,1130) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  140 WRITE(6,1140) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  150 WRITE(6,1150) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  160 WRITE(6,1160) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  170 WRITE(6,1170) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  180 WRITE(6,1180) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
      GO TO 200
  190 WRITE(6,1190) TEMP1,TEMP2,(PURCH(J,I),J=1,NV)
  200 CONTINUE
 1110 FORMAT(1H ,A6,2X,A2,2X,  1H*,2X,F8.3,2X  / 13X,  1H*,12X  )
 1120 FORMAT(1H ,A6,2X,A2,2X,2(1H*,2X,F8.3,2X) / 13X,2(1H*,12X) )
 1130 FORMAT(1H ,A6,2X,A2,2X,3(1H*,2X,F8.3,2X) / 13X,3(1H*,12X) )
 1140 FORMAT(1H ,A6,2X,A2,2X,4(1H*,2X,F8.3,2X) / 13X,4(1H*,12X) )
 1150 FORMAT(1H ,A6,2X,A2,2X,5(1H*,2X,F8.3,2X) / 13X,5(1H*,12X) )
 1160 FORMAT(1H ,A6,2X,A2,2X,6(1H*,2X,F8.3,2X) / 13X,6(1H*,12X) )
 1170 FORMAT(1H ,A6,2X,A2,2X,7(1H*,2X,F8.3,2X) / 13X,7(1H*,12X) )
 1180 FORMAT(1H ,A6,2X,A2,2X,8(1H*,2X,F8.3,2X) / 13X,8(1H*,12X) )
 1190 FORMAT(1H ,A6,2X,A2,2X,9(1H*,2X,F8.3,2X) / 13X,9(1H*,12X) )
C
C FIRST PART OF THIS SUBROUTINE OUTPUT INFORMATION CONCERNING
C EQUIPMENT PURCHASES DURING EACH PERIOD .....
C NEXT SECTION OUTPUTS RESOURCES STORED
C
      WRITE(6,3000) (TITLE(I),I=1,4)
      M=3
      GO TO 5
  305 IF(M.EQ.2) GO TO 205
      N=0
      DO 400 I=1,NP
      K=IYR(I)-INYR+1
      IF(K.LE.0) GO TO 400
      N=N+1
      GO TO (310,320,330,340,350,360,370,380,390),NV
  310 WRITE(6,1110)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  320 WRITE(6,1120)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  330 WRITE(6,1130)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  340 WRITE(6,1140)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  350 WRITE(6,1150)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  360 WRITE(6,1160)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  370 WRITE(6,1170)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  380 WRITE(6,1180)PERTOD,CODE(N),(STOR(J,I),J=1,NV)
      GO TO 400
  390 WRITE(6,1190)PERIOD,CODE(N),(STOR(J,I),J=1,NV)
  400 CONTINUE
 3000 FORMAT(1H1,15X,4A10 / 1H-,*STORED RESOURCES* )
C
C REMAINING PART WILL OUTPUT THE TOTAL AMOUNT USED
C DURING EACH PERIOD
C
      WRITE(6,2000)(TITLE(I),I=1,4)
      M=2
      GO TO 5
  205 N=0
      DO 300 I=1,NP
      K=IYR(I) -INYR+1
      IF(K.LE.0) GO TO 300
      DO 206 J=1,NV
  206 EXIST(J,K)=EXIST(J,K)-STOR(J,I)
      N=N+1
      GO TO (210,220,230,240,250,260,270,280,290),NV
  210 WRITE(6,1110) PERIOD,CODE(N),(EXTST(J,K),J=1,NV)
      GO TO 300
  220 WRITE(6,1120) PERTOD,CODE(N),(EXTST(J,K),J=1,NV)
      GO TO 300
  230 WRITE(6,1130) PERIOD,CODE(N),(EXIST(J,K),J=1,NV)
      GO TO 300
  240 WRITE(6,1140) PERIOD,CODE(N),(EXIST(J,K),J=1,NV)
      GO TO 300
  250 WRITE(6,1150) PERIOD,CODE(N),(EXIST(J,K),J=1,NV)
      GO TO 300
  260 WRITE(6,1160) PERIOD,CODE(N),(EXIST(J,K),J=1,NV)
      GO TO 300
  270 WRITE(6,1170) PERIOD,CODE(N),(EXIST(J,K),J=1,NV)
     .GO TO 300
  280 WRITE(6,1180) PERIOD,CODE(N),(EXTST(J,K),J=1,NV)
      GO TO 300
  290 WRITE(6,1190) PERIOD,CODE(N),(EXTST(J,K),J=1,NV)
  300 CONTINUE
      RETURN
 2000 FORMAT(1H1,15X,4A10 / 1H-,*TOTAL RESOURCES USED* )
      END
      SUBROUTINE SETUP
      COMMON /VECSTG/ VNAME(10),C,LENP,VLIFE(10),INH(10,16),
     *  VCOST(10,5),NAMEN(10),COSTS(30,3)
      COMMON /BASICS/ CHAR(5000,4),CODE(20),PER(10),IYR(10),LYR(10)
      COMMON /OUTS/ QANDM(20),SALE(20),SAVE(20),EXIST(10,20),
     *  PURCH(10,20),STOR(10,20),SALV(10,20),PROC(20),PROT(20)
      COMMON /PARAMS/ PDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST
      DIMENSION TEMP(4)
      DATA IVT,IPT,IED/8HVEHICLE  ,8HPERIOD   ,8HENDTABLE /
      NVR=0
      NPT=0
      READ(5,1000)FNAME,INYR,LAST,NV,NT,NP
 1000 FORMAT(A8,2X,6I5)
   10 READ(5,1000)ITABLE
      IF(ITABLE.EQ.IVT) GO TO 20
      IF(ITABLE.EQ.IPT) GO TO 60
      IF(ITABLE.EQ.IED) GO TO 100
      WRITE(6,2000)ITABLE
 2000 FORMAT(1H-,A8,* IS NOT RECOGNIZED BY SETUP*)
      STOP
   20 NVR=NVR+1
      READ(5,4000)VNAME(NVR),VLIFE(NVR)
      READ(5,1070)(VCOST(NVR,I),I=1,5)
 1070 FORMAT(5F10.2)
      GO TO 10
   60 NPT=NPT+1
      READ(5,1140)  IYR(NPT),LYR(NPT),PER(NPT),PROT(NPT)
 1140 FORMAT(I4,I5,1X,A2,F8.2)
      GO TO 10
 4000 FORMAT(A8,11X,I2)
100     CONTINUE
        PROT(NPT+1) = 0.
110     CONTINUE
      READ(9,3000)  I,(TEMP(J),J=1,4)
 3000 FORMAT(I5,4X,A1,3A2)
      IF(EOF,9)200,150
  150 DO 160 J=1,4
  160 CHAR(I,J)=TEMP(J)
        GO TO 110
  200 RETURN
      END
      SUBROUTINE VALUES(N,ISTART,IEND,VAL)
      COMMON /VECSTG/ VNAMF(10),C,LENP,VLIFE(10),INH(10,16),
     *  VCOST(10,5),NAMEN(10),COSTS(30,3)
      COMMON /BASICS/ CHAR(5000,4),CODE(20),PEP(10),IYR(10),LYR(10)
      COMMON /OUTS/ OANDM(20),SALE(20),SAVE(20),EXIST(10,20),
     *  PURCH(10,20),STOR(10,20),SALV(10,20),PROC(20),PROT(20)
      COMMON /PARAMS/ RDTOT,INYR,LAST,NV,NP,TOT,TITLE(4),COST
      CALL YRCOST(N)
      I=ISTAPT-INYR
      II=TEND-ISTAPT+1
      K=1
      DO 10 J=1,II
      I=I+1
      IF(I.LE.0) GO TO 10
      OANDM(I)=OANDM(I)+COSTS(J,K)*VAL
      EXIST(N,I)=EXIST(N,I)+VAL
   10 CONTINUE
      I=I+1
      K=K+1
      IF(IEND.EQ.LAST) K=K+1
      SALE(I)=COSTS(J,K)*VAL+SALE(I)
      SALV(N,I)=VAL+SALV(N,I)
      RETURN
      END
      SUBROUTINE YRCOST(J)
C  A SUBROUTINE TO COMPUTE THE OPERATING, SALVAGE, AND TRUNCATION
C  COSTS YEAR BY YEAR.  ALSO THE YEARLY MOTHBALLING SAVING IS COMPUTED.
      COMMON /VECSTG/ VNAME(10), C,LENP, VLIFE(10), TNH(10,16),
     * VCOST(10,5), NAMEN(10), COSTS(30,3)
      INTEGER VNAME,VLIFE
C  ASSUME THE OPERATING AND MAINTANCE COST INCREASES AT R*100 PER-CENT
C  A YEAR (NOT A COMPOUND RATE INCEASE)
      R=0.0
C
C  LET X= THE 1ST YEAR O. AND M. COST.  THEN
C                                   X+(1+R)*X+(+2*R)*X+...+(1+9*R)*X=VCOST(J,2)
      X= VCOST(J,2)/(10.0 + 45.0*R)
C  ASSUME NO PERIOD IS LONGER THAN 6 YEARS.
      IB=VLIFE(J) +10
      DO 10 I=1,IB
      COSTS(I,1)=(1.0 + FLOAT(I-1)*R)*X*(VCOST(J,4)**(I-1))
   10 CONTINUE
C
C  ASSUME THE SALVAGE VALUE OF A VEHICLE AFTER I YEARS OF SERVICE IS
C  (ALPHA)**I *PURCHASE COST.
      ALPHA=0.5
      Y=VCOST(J,1)
      DO 20 I=1,IB
      Y= ALPHA*Y
      COSTS(I,2)=Y
   20 CONTINUE
C
C           ASSUME TRUNCATION AFTER IYEARS OF SERVICE IS
C                  (VLIFE-I)*(PURCHASE COST)/VLIFE
C
      Y=VCOST(J,1)/VLIFE(J)
      DO 30 I=1,IB
      IX=VLIFE(J)-I
      IF (IX.LT.0) IX=0
      COSTS(I,3)=IX*Y
   30 CONTINUE
      RETURN
      ENTRY MOTH
C  ASSSUME THE MOTHBALLING SAVING IS R1*100 PER CENT OF THE FIRST YEAR COST-X
      R1=0.90
C     C=0
C     DO 546 IL=1,LENP
C 546 C=C-0.1*R1*VCOST(J,2)*VCOST(J,4)**(IL-1)
C     C =-X * R1
      C=-VCOST(J,2)/(10.0 + 45.0*R) * R1
      RETURN
      END
```

APPENDIX E

ERROR MESSAGES

ERROR MESSAGES FROM MATRIX GENERATOR

"... is not a table name."

This message indicates that the input deck is not properly constructed since the program has read a card which should have been a header card but was not. The location of the error can be narrowed down by checking the output listing to see which tables have been correctly read. This error terminates execution.

"Vehicle name ... not defined in a vehicle."

This message is output when a task table is being read. It indicates either that the vehicle name is misspelled or located improperly on the card, or that the task table has preceded the vehicle table, if the vehicle table exists. This error also terminates execution.

"The period tables are out of order."

This message indicates that the first year of the period just read was not equal to one plus the last year of the last period read. This can be caused by improper sequencing or improper definition of the periods. This error will cause execution to terminate.

"Warning--the number of tables input was not the expected number."

This message does not terminate execution, but does indicate that there was a difference between the number of tables indicated on the title card and the number actually read by the program.

"Incorrectly read file ... columns read as ..."
"The M-1 column was ..., unable to find RHS mark."
"Reached EOF while writing column ... and row ..."

These three error messages all refer to errors encountered when trying to convert the MPS360 file to the file for BBCAV2. If one of these errors occurs, a major problem exists within the program. As a result these prevent the creation of the BBCAV2 file, but allow the program to execute to completion to give the analyst the most information possible about the problem.

ERROR MESSAGES FROM MAIN PROGRAM

"BLIST size exceeded"

BLIST is the array used for storing nodes of the branching tree. It is presently dimensioned to handle 25 nodes. When there exist more than 25 nodes which have been defined but have not been evaluated, this message is generated. It indicates that this particular problem is converging very slowly, and if one desires an accurate answer then the arrays EKBL, PSIGL, NXBL, XNXBL, and BLIST should be enlarged. This error causes the system to print out the best solution found and proceed to the next problem.

"Time is up . . . cycling to next problem."
"Have solved max. no. of LP probs."

These two messages inform one that the solution which is output is not necessarily optimal but was caused by one of the input parameters. The first message indicates a violation of the time indicated by the second field of the real parameter card. The second message is a result of reaching the limit on the number of nodes (LP problems) which is set in the last field of the integer parameter card.

"Premature EOF on a matrix tape at column . . ."

This message indicates that the size of the tape file does not correspond with the size indicated on the integer parameter card. It also gives an indication of the size of the tape file for comparison against that which was input. This error terminates execution of the program.

"KFX = 0 in GETPHI"
"Invalid NOES in GETASQ = 0"
"Invalid KCX = 0 in GETC"

These three error messages all indicate that an invalid parameter value has been passed from some routine to one of these listed above. These messages were used primarily for debugging and should not appear in normal operations. If they do, it indicates an error exists in the program code somewhere. These messages all terminate execution.

"LP - insufficient space allocated in NWAJ"

This fatal error is a result of having a matrix in excess of 100 rows, thus giving an inverse too large to be stored in the array AJ (11000). To correct this condition set AJ (-----) to an appropriate size for the problem, and set NWAJ equal to that value. If this error occurs, many other arrays may also need to be redimensioned to insure proper storage.

"SETUP--row type error--out of range"

This statement indicates the row type indicator exceeds 4 and therefore the row cannot be defined. For this problem structure the only valid row types are 0 for equality, 3 for free, and 4 for generalized upper bound. This error should not occur, since the vector IROWTP is set by the matrix generator. This error will terminate execution.

"PRIMAL--too many reject vectors"

This message implies that more than 100 columns have been rejected for degeneracy reasons. This is fatal if the LP is in the infeasible phase, and causes an optimal solution to be assumed otherwise.

"IO--column not located in NT reads"
"Row--key not in core"
"Insert cannot find rejected column"
"PIVOT--PIVOT less than PIVTOL"
"KEYCH--essential packet no basic column"

These five error messages are used exclusively for debugging, and should not occur in normal operation. The general cause for this is that some section of core has been overwritten accidently.

"PIVOT dropped column . . ."

This message indicates that a column was removed from the basis during the inversion process. This occurs when the input basis is not feasible, and when numerical errors have caused the current basis to "drift" out of the feasible region.

ERROR MESSAGES FROM REPORT GENERATOR

". . . is not recognized by SETUP."

The routine SETUP has encountered an error in the input deck while attempting to read a table name. This error terminates execution.

"Error in interpretation of . . ."

The program has been unable to determine the meaning of the seven-character code indicated in the message. If the code is a valid one (one of the forms shown in Fig. 11), then there is probably an error in the period descriptions of the input deck. There is also possibility of other errors in the input deck or, as a last resort, of errors in the reference list file. If the code is not a valid one, then the error must be in the reference list. This error also terminates execution.

GLOSSARY

This section contains the mnemonic definitions for all three programs—GENLCP, BBCAV2, and REPGEN. It is arranged into two major sections. The first section lists the mnemonics in labeled common—then the local variables contained within each subroutine, for each of the three programs, respectively. The second section is an overall alphabetical listing for handy reference. Note that in this listing, the same mnemonic may have two or more meanings. Each entry is identified here as a local or global variable, and is cross-referenced to the first section. Use of the two sections, in conjunction, should eliminate any ambiguity.

SECTION 1..................G-2

SECTION 2..................G-25

Section I

GENLCP CODING DEFINITIONS

COMMON/VECSTG/

VNAME (10)	- vehicle names
C	- temporary storage for cost data
LENP	- length of period
VLIFE (10)	- maximum life of resource (vehicle)
INH (10, 16)	- number of each type resource inherited from each year
VCOST (10,5)	- cost data for each resource
NAMEN (10)	- pointers for numbering resources
COST (30, 3)	- yearly operating, salvage and truncation costs

COMMON/ALTSTG/

ALTER (288,9)	- array used for eliminating infeasible alternatives from tasks
YAVL (10)	- year resource first available

COMMON/TSKSTG/

U (7, 288, 9)	- array of task alternatives
NTSK (9)	- number of alternatives in task

COMMON/PRDSTG/

NPERYR (10, 3)	- first and last year of period and number of tasks in period
NPTASK (10, 9)	- ID number of each task in period
PTASK (10, 9)	- multiplicative factor for all values in associated task for each period

LOCAL VARIABLES

GENLCP

ALPHA	- temporary storage for attrition
AU (16)	- temporary storage for alternatives

BUDG (10)	limit on procurement expenditures in each period
CMAX	temporary cost storage for ordered resources
FNAME	file name
IHVN (10)	pointers for inherited vehicles
INHYRS	number of years from which vehicles are inherited
ITABLE	temporary storage for table name
LIFER	temporary storage for remaining useful life of a vehicle
LY	last year of problem
MAXL	temporary storage for vehicle life
MCOL	number of columns in matrix
NAMES (10)	temporary pointers
NINHP	number of inherited periods
NIV	number of inherited vehicle types
NL (10)	temporary storage used in formatting output
NN (10)	
NPP	number of periods
NPT	number of period tables read
NRD	number of vehicles having R&D
NROW	number of rows in matrix
NT	number of tasks
NTR	number of task tables read
NV	number of vehicle types
NVEHU (10)	indicates if vehicle used in period
NVR	number of vehicle tables read
NYR	temporary storage for last year of period
ONE	"1.0"
ONEM	"-1.0"

SY	- start year of problem
UB (10)	- calculated upper bounds on resources
UMAX	- temporary storage for greatest quantity of a specific vehicle which might be used in a task
YEARS (21)	- stores inherited years
YRINT (20)	- scale factor for all tasks in period

YRCOST

ALPHA	- rate of decrease in salvage value
R	- rate of increase in operating cost
R1	- portion of operating cost refunded for mothballing resource

YINTERP

JSUB (10)	- pointers for vehicle subscripts
VMIN	- temporary storage for minimum quantity of vehicles which can be used for a task

MATFILL

C	- "COLUMNS"
CNAME	- column name for which RVAL is being derived
CTEMP	- temporary storage for column name
IROWTP (100)	- indicates row type; all set to zero except generalized upper bound rows which are set to 4
ITEMP	- temporary storage for first letter of RNAME
R	- "RHS"
RNAME (120)	- row names
RTEMP	- temporary storage for row names
RVAL (100)	- vector of values in each row for a specific column
VAL	- temporary storage for value of specific row and column

BBCAV2 CODING DEFINITIONS

COMMON/CV1/

IP (12) - storage for input parameters on integer parameter card

RP (12) - storage for real parameters, first four locations are for input from real parameter card, rest are temporary storage

TMP (10) - temporary storage

COMMON/CV2/

T (100, 10) - storage for columns of matrix associated with nonlinear variables

BO (100) - right-hand-side vector

BLO (10) - set of lower bounds on nonlinear variables

ULO (10) - set of upper bounds on nonlinear variables

CO (10) - vector for linear approximation for nonlinear cost functions

COMMON/CV3/

M - number of rows in matrix

N - number of columns in matrix

NCF - number of nonlinear variables

PHIT - cost of a nonlinear solution

UZ - cost of best nonlinear solution

USP - $UZ\ (1 + \epsilon)^{-1}$

USM - $UZ\ (1 - \epsilon)^{-1}$

EKO - cost associated with the lower bounds of the node

MPLUS - number of rows in the matrix including the cost row (M + 1)

COMMON/CV4/

IX (110) - columns in basic solution

X (110) - values associated with columns in IX

IXZ (110) - columns in best solution

XZ (110) - values associated with columns in IXZ

XCON (10) - stores values found in X which are associated with the nonlinear variables

COST - cost of the solution returned from the LP

COMMON/CV5/

SIGMA (100, 4) - stores information which defines the current node

TSIG - temporary storage associated with EKO

LSTMAX - maximum length which the branching list has achieved

COMMON/CV7/

NPHASE - stores LP phase code

NF1 - signifies feasible solution when set equal to 1

CFX - no longer used

IOPT - used to flag unbounded solution

NOP - node number

NOPS - nodes solved

NEWXZ - flags when new best solution found and should be output

COMMON/CV8/

NXBK - index of branching variable

XK - value of branching variable

NOBOL - number of nodes on list

EKBL (25) - EKO value associated with each node on the list

COMMON/CV9/

PSIGL (25) - lower bound associated with each node on list

NXBL (25) - index of branching variable for each node

XNXBL (25) - value of branching variable for each node

BLIST (25, 131)- branching list; contains right-hand-side vector, plus upper bounds, lower bounds, and linear cost approximations for nonlinear variables

COMMON/TMX/

TMO - time SET was called

EXT - time when time limit on problem will expire

COMMON/CORE/

AJ (5000) - columns in core plus basis inverse

JA (101) - in-core column disc indices

JAK (101) - dummy storage area

JAREJ (101) - set to 1 when corresponding in-core column rejected

COMMON/PARAMS/

TMAX - maximum time before MAPOUT

ITNINV - iteration of next invert

INVF - invert frequency

K1 - not used

K2 - not used

K3 - output control parameter

K4 - XCHECK control parameter

K5 - maximum LP iterations before MAPOUT

COMMON/INPUT/

INPUT - file containing input matrix

INPUTM - number of rows in matrix

INPUTN - number of columns in matrix

COMMON/FILES/

IA1 - disc file for matrix less GUB rows

IA2 - disc file for packed matrix less GUB rows

IMAP - file for starting and terminating basis

COMMON/STATE/

IROW - current selected row

ITRN	- iteration count
JCOL	- current selected column
JOUT	- rejected column index
JPOS	- selected column index
NDJS	- number of negative DJ's
NPIF	- number of primal infeasibilities
NREJ	- number of rejected in-core columns

COMMON/LIMS/

JNCORE	- number of columns in core
MAXTRY	- maximum number of in-core iterations
NCRMAX	- maximum number of columns which fit in core
NSCAN	- number of disc reads
NTRY	- number of in-core iterations

COMMON/IXX/

IX (100)	- indices of solution columns

COMMON/XX/

X (100)	- values of solution columns

COMMON/TOLS/

CTOL	- cost tolerance for infeasibility
DERTOL	- dual error tolerance; not used
DJTOL	- DJ tolerance
PERTOL	- primal error tolerance
PIVTOL	- pivot tolerance
ZERO	- smallest recognized number

COMMON/BASIS/

IBASIS (101)	- basic columns for non-GUB rows
KEYS (101)	- storage of GUB key columns

COMMON/DJS/

DJ (100)	- values of current in-core DJ's

COMMON/MOVES/

BNDJ	- value of current column bound
DMAX	- largest DJ value stored
DUALER	- dual error; unused
PRMLER	- primal error; unused
THETA	- step chosen by ROW, adjusted in PRIMAL

COMMON/I/

IC	- current cost row
ICOST	- user's cost row
IPHASE	- current LP phase
IPI	- current location of PI vector in basis
JRHS	- user's input RHS
L	- number of GUB rows
M	- number of active interval rows
MC	- last logical column
MPL	- M plus L
NT	- total number of columns (MC + INPUTN)

COMMON/A/

ALPHA (101)	- work space, usually current column inverse

COMMON/B/

BETA (101)	- work space usually values of basic and key variables

COMMON/C/

GAMMA (101)	- not used

COMMON/D/

DELTA (101)	- not used

COMMON/ROWTYP/

IROWTYP (101) - user's input row types

COMMON/NAMES/

NAME (600) - state of each variable or column

COMMON/BOUNDS/

BOUNDS (100) - values of upper bounds

IBDS (100) - column indices of bound columns

NBDS - number of bounds

COMMON/RHS/

RHS (100) - stores user's current right-hand-side

LOCAL VARIABLES

BBCAV2 and BOX1

BLT (10) - temporary storage for BLO

BBK - lower bound on branching variable

BBK2 - value of branching variable

COST1 - solution cost for lower branch

COST2 - solution cost for upper branch

CT (10) - temporary storage for CO

EPSI - epsilon value from real parameter card

ESIG - temporary storage for EKO

ICOL - temporary storage for column index

INDIC - indicates which branch (upper or lower) is being solved

LSTFRE (25) - gives locations of storage areas on the branching list which are vacant

MNC - the negative of NCF

MNX - the negative of N

NCF1 - NCF

NCF4 - NORA + 3 * NCF

NFREE - number of gaps (empty location between two filled locations) in the BLIST

NMIN - index of the lowest bound on the BLIST, or N-1, depending on where it is used

NOL - index for storage on BLIST

NORA - M

NXB - temporary storage for next branching variable

PH1 and PH2 - temporary storage of values from GETPHI

PMIN - value of lowest bound on BLIST

TSTO (130) - temporary storage

TITLE (4) - alphanumeric title of problem

UBK - upper bound on branching variable

UBK2 - difference between upper bound and value for branching variable

ULT (10) - temporary storage for ULO

US - temporary storage for USP

INITA

AJ (100) - temporary storage for column of matrix

DUM1 and DUM2 - temporary storage for reading unused sections of tape

READIN

NC - number of basis cards to be read from input

TIMEC

SECS - actual CPU clock time

XX - elapsed time on problem

GETASQ

TEMP - location used while swapping contents of two locations in an array

NEM1 - number of elements in an array minus one

GETC

DIF - difference between upper and lower bound for a variable

FX1 (10) - cost function values for lower bounds

FX2 (10) - cost function values for upper bounds

ICX - number of variables for which cost slopes are to be derived

NXBRN

BLT (10) - temporary storage for BLO

CT (10) - temporary storage for CO

YT (10) - differences between solution point and lower bounds

FX1 (10) - cost function values for lower bounds

FX2 (10) - cost function values for solution point

DIF (10) - differences between cost functions and linear approximations

NDX (10) - indices of nonlinear variables

NFX - the negative of NCF

XT (10) - solution values for nonlinear variables

PRESET AND PARAMS

IBMAX - maximum number of nodes which may be stored on BLIST

JBMAX - maximum number of words of information which may be stored for each node in BLIST

NORA - number of rows in the matrix including the objective function

LP

B (100) - basis inverse stored by rows

IORG - origin of basis inverse

MROWS - user's number of rows

NCHGS - user's number of bound columns

NCOLS	user's number of columns
NWAJ	storage dimension of the array AJ

SETUP

ID	local row type being processed
IKOST	temporary storage of user's cost row
INON	temporary number of non-GUB rows found
PKT1	temporary count of GUB row packet columns
PKT	actual GUB row column being processed

IO

ALPHA	column to be written or read
B	address of origin of basis inverse
JCOL	core position of column being read
JNT	index of columns read
KEY	index of key column to be located
KOL1	last column read on file IA1
KOL2	last column read on file IA2
KOL	column to be located on either file or packet number of desired key
NAAM	not used
NAME	column name, or position in core to which column is read
PACK (100)	temporary storage of packed column
ZS	parameter used to pack coefficients
Z	parameter used to pack index of coefficient

MAPIN

ATBND	"ATBND"
BASIC	"BASIC"
BNDJ	value of bound
B	origin of basis inverse

CARD (8)	- image of map card
ENDER	- "END"
ID	- column number from map card
INVERSE	- "INVERS"
KEE	- "KEY"
MM	- number of elements in basis inverse
NAMES (5)	- column indices from map card
NULL	- "NULL"
PKT	- storage of column packet
ROWS	- "ROWS"
TYPE1	- first word on map card
TYPE2	- second work on map card

MAPOUT

IBAS	- count of basis variables
IBND	- count of bound variables
IKEY	- count of key variables
INLL	- count of null variables
JCOL	- user's column index of column processed
JNCORE	- number of columns in core
MAPBAS (100)	- basic column indices
MAPBND (100)	- bound column indices
MAPKEY (1000)	- key column indices
MAPNLL (10)	- null column indices
MM	- number of elements in basis inverse
MP1	- M plus 1

INVERT

ATBND	- column type
BASIC	- column type

BNDJ	- cound on current column
B	- origin of basis inverse
IORG	- origin of first element in B
ITYPE	- row type
JNT	- current column index
JORG	- origin in AJ to which column is read
JTYPE	- variable type
KORG	- origin in AJ to which key column is read
PKTO	- GUB packet number of column in AJ (KORG)
PKT	- GUB packet number of column being processed

FEASCH

BNDJ	- bound on current column
B	- basis inverse
IORG	- origin of any row in B
JPKT	- GUB packet of current column
KEY	- switch to return key processing to key loop
NB	- number of basic variables in a packet
SUMIE	- sum of infeasibilities
SUM	- value of variable before feasibility adjustment

PRIMAL

BASIC	- column type
B	- basis inverse
EPSI	- value of new basic variable
ITYPE	- type of step to be used
JOUTPK	- GUB packet of column rejected
JPOSPK	- GUB packet of column entering
NBVPKT	- number of basis variable in selected GUB row

NPEGLM	- maximum rejection due to degeneracy
NDEG	- number of degeneracy rejections
NEWROW	- row for column changing from key to basic

STATUS

ATBND	- state of a column
BASIC	- state of a column
B	- basis inverse
COST	- value of current objective function
JNSCAN	- columns in core + 1000 times number of rewinds of file IA1
JNTO	- index of last column read from disc
JNT	- last column read from disc (if MNTRY = 0)
MNTRY	- number of in-core iterations
NCOLS	- number of columns read from disc file IA1
NJOUT	- name code of column to be rejected
NOTE (4)	- 40 character comment
X	- elapsed CPU seconds

ROW

BASIC	- state of a column
B	- basis inverse
IB	- basic column index
IORG	- origin of basis inverse
IROW	- row calling parameter, row of zero
ITYPE	- type of step; 1-unbounded, 2-column to zero, 3-column to bound
JCOL	- core index of selected column
JORG	- core origin of selected column
JOUT	- column to be rejected
JPKT	- GUB packet of column selected

JPOS	- disc index of column selected
KORG	- origin of KEY column for packet JPKT
STEP	- step to current row
THETA	- best feasible step

COLUMN

ATBND	- logical column state
BASIC	- logical column state
B	- basis inverse
JCOL	- core position of selected column
JKEY	- core position of key for JCOL (if in GUB row)
JORG	- origin of a row in B
JPKTO	- current stored GUB key packet
JPKT	- GUB packet of new column
JTYPE	- type of column selected
KORG	- origin of KEY in AJ
NCORE	- number of columns in core
NDJST	- number of negative DJ's from disk read
NULL	- column state
PIKEY	- PJ value for current KEY JPKT

CHECK

ATBND	- state of column
BASIC	- state of column
B	- basis inverse
DJ	- current column sensitivity
JCOUNT	- count of columns processed
JFBCH	- number of columns, checked in current batch
JNT	- index of current column

JORG	- origin in AJ to which columns are read
JTYPE	- type of column being processed
KORG	- origin of key column in AJ
NBCH	- number of columns in batch
NFBCH	- number of columns retained from batch
PIKEY	- DJ for current key at KORG
PKTO	- packet of current key
PKT	- packet of new column, JNT

INSERT

B	- basis inverse
DJ	- DJ for column to be stored
DMAX	- largest DJ of stored columns
D (15)	- DJ's of stored columns
ID (15)	- indices of stored columns
JORG	- origin of vacancy for column in AJ
JPOSR	- disc index of column to be rejected
JPOS	- disc index of column to be stored
JREJ	- origin of rejected column in AJ
NPBCH	- number of columns to be saved from batch
N	- number of columns currently saved

KEYCH

B	- basis inverse
IB	- disc index of basic column for current row
IORG	- origin of a row in B
IROW	- row to which key column is shifted when made basic
JCOLPK	- GUB packet of column being moved from KEY
JCOL	- column to be moved

JKEY	- candidate key column
JORG	- origin of a row in B
MPK	- row of column which was KEY
SUM	- temporary storage

PIVOT

ALPHA	- column to be pivoted into basis
B	- basis inverse
DIVOT	- candidate pivot while searching for best
IORG	- origin of pivot row in B
IROW	- pivot row
JORG	- origin of a row in B
JP	- basic column for a row
PIV	- pivot used

SETBND

I	- input disk column index
J	- absolute value of I
K	- new state

DOT

DOT	- double precision inner product of X and Y
DOTS	- single precision inner product of X and Y
M	- vector dimension
SUM	- double precision accumulator
X	- input vector
Y	- input vector

BOUND

BOUND	- value of column bound (or 10**70)
IB	- bound index in IBDS
J	- input disc column index

KEYFIND

I	- dummy variable
JAJ	- potential column's in-core position
JPKT	- GUB packet number for column
JTYPE	- column type
KEYFIND	- position of key found
KEY	- column number of key to be located
PKT	- GUB packet of desired key

ESCAPE

AALPHA	- "ALPHA"
ABASIS	- "BASIS"
ABETA	- "BETA"
ADELTA	- "DELTA"
ADJ	- "DJ"
AGAMMA	- "GAMMA"
AJAREJ	- "JAREJ"
AJA	- "JA"
AKEY	- "KEY"
ANAME	- "NAME "
B	- basis inverse

XCHECK

ATBND	- logical column state
AT	- dummy
BASIC	- logical column state
B	- basis inverse
CALLER	- calling name
IORG	- origin of a row in basis inverse

JAJ	- disk index of an in-core column
JEND	- origin of vacant work space in AJ
JORG	- origin of a column in AJ
J1	- first column in column printout
J2	- last column in column printout

REPGEN CODING DEFINITIONS

COMMON/VECSTG/

VNAME (10) - stores resource names

C - temporary storage location used in calculating savings from resource storage

LENP - length of period under consideration

VLIFE (10) - expected resource life

INH (10, 16) - not used

VCOST (10, 5) - the five costs associated with each resource are stored in this array; in order, they are salvage and truncation, operating, R&D, retention rate, and procurement. (Explained in detail in matrix generator description.)

NAMEN (10) - not used

COSTS (30, 3) - cost of operating (1), selling (2), or truncating (3), a resource in the 1st thru 30th year of its life

COMMON/BASICS/

CHAR (5000, 4) - storage of column names which have been broken down into their four meaningful parts

CODE (20) - storage of the numbers 1 - 20 in two digit alphanumeric form

PER (10) - pointers for two digit, alphanumeric code for periods

IYR (10) - initial year of each period

LYR (10) - last year of each period

COMMON/OUTS/

OANDM (20) - operating cost for each year

SALE (20) - salvage or truncation value for each year

SAVE (20) - savings from resource storage for each year

EXIST (10, 20) - number of each type resource available in each year

PURCH (10, 20) - number of each type resource purchased in each year

STOR (10, 20) - number of each type resource stored in each year

SALV (10, 20) - number of each type resource disposed of at end of each year

PROC (20) - procurement funds spent during each period

PROT (20) - procurement funds available during each period

COMMON/PARAMS/

RDTOT - total R&D expenditures

INYR - initial year of problem

LAST - last year of problem

NV - number of resource types

NP - number of subperiods

TOT - number of subperiods plus 1

TITLE (4) - name of specific solution

COST - total cost of solution

LOCAL VARIABLES

SETUP

FNAME - problem title (not used)

IED - "ENDTABLE"

IPT - "PERIOD"

ITABLE - table name

IVT - "VEHICLE"

NPT - period tables read in

NT - number of tasks (not used)

NVR - number of resources read in

TEMP (4) - temporary storage for column names

INSOLN

BLANK - "   "

IEND - last year of resource existance

IND - column number temporary storage

ISTART - first year of resource existance

S - "S"

VAL - column value temporary storage

W - "W"

X - "X"

<u>CINFD</u>

PERIOD - "PERIOD"

ONE - "01"

SUM - total cost for a period

SUMT - total cost for all periods

TCOST - temporary storage for total procurement

TOTAL - "TOTAL"

TPROC - correction factor for procurement

<u>PINFO</u>

TEMP1 -
TEMP2 - } temporary storage locations for alphanumeric output

BLANK - "   "

PERIOD - "PERIOD"

TOTAL - "TOTAL"

```
AALPHA - 'ALPHA'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
ABASIS - 'BASIS'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
ABETA - 'BETA'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
ADELTA - 'DELTA'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
ADJ - 'DJ'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
AGAMMA - 'GAMMA'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
AJAREJ - 'JAREJ'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
AJA - 'JA'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
AJ(5000) - COLUMNS IN CORE PLUS BASIS INVERSE
                    (GLOBAL - MAIN PROGRAM'S COMMON / CORE / )
AJ(100) - TEMPORARY STORAGE FOR COLUMN OF MATRIX
                    ( LOCAL - MAIN PROGRAM'S INITA )
AKEY - 'KEY'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
ALPHA(101) - WORK SPACE, USUALLY CURRENT COLUMN INVERSE
                    (GLOBAL - MAIN PROGRAM'S COMMON / A / )
ALPHA - COLUMN TO BE WRITTEN OR READ
                    ( LOCAL - MAIN PROGRAM'S IO )
ALPHA - TEMP STORAGE FOR ATTRITION
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
ALPHA - RATE OF DECREASE IN SALVAGE VALUE
                    ( LOCAL - MATRIX GENERATOR'S YRCOST )
ALPHA - COLUMN TO BE PIVOTED INTO BASIS
                    ( LOCAL - MAIN PROGRAM'S PIVOT )
ALTER (288, 9) - ARRAY USED FOR ELIMINATING INFEASIBLE ALTERNATIVES FROM TASK
                    (GLOBAL - MATRIX GENERATOR'S COMMON / ALTSTG / )
ANAME - 'NAME'
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
AT - DUMMY
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
ATBND - 'ATBND'
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
ATBND - COLUMN TYPE
                    ( LOCAL - MAIN PROGRAM'S INVERT )
ATBND - LOGICAL COLUMN STATE
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
                    ( LOCAL - MAIN PROGRAM'S CHECK )
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
                    ( LOCAL - MAIN PROGRAM'S STATUS )
AU (16) - TEMP. STORAGE FOR ALTERNATIVES
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
```

```
B - BASIS INVERSE
                    ( LOCAL - MAIN PROGRAM'S LP )
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
                    ( LOCAL - MAIN PROGRAM'S ESCAPE )
                    ( LOCAL - MAIN PROGRAM'S PIVOT )
                    ( LOCAL - MAIN PROGRAM'S KEYCH )
                    ( LOCAL - MAIN PROGRAM'S INSERT )
                    ( LOCAL - MAIN PROGRAM'S CHECK )
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
                    ( LOCAL - MAIN PROGRAM'S ROW )
                    ( LOCAL - MAIN PROGRAM'S STATUS )
                    ( LOCAL - MAIN PROGRAM'S PRIMAL )
                    ( LOCAL - MAIN PROGRAM'S FEASCH )
                    ( LOCAL - MAIN PROGRAM'S INVERT )
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
                    ( LOCAL - MAIN PROGRAM'S IO )
BASIC - 'BASIC'
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
BASIC - COLUMN TYPE
                    ( LOCAL - MAIN PROGRAM'S INVERT )
                    ( LOCAL - MAIN PROGRAM'S PRIMAL )
BASIC - LOGICAL COLUMN STATE
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
                    ( LOCAL - MAIN PROGRAM'S CHECK )
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
                    ( LOCAL - MAIN PROGRAM'S STATUS )
                    ( LOCAL - MAIN PROGRAM'S ROW )
BBK - LOWER BOUND ON BRANCHING VARIABLE
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
BBK2 - VALUE OF BRANCHING VARIABLE
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
BETA(101) - WORK SPACE, USUALLY VALUES OF BASIC AND KEY VARIABLES
                    (GLOBAL - MAIN PROGRAM'S COMMON / B / )
BLANK - '       '
                    ( LOCAL - REPORT GENERATOR'S INSOLN )
                    ( LOCAL - REPORT GENERATOR'S PINFO )
BLIST(25,131) - BRANCHING LIST, CONTAINS RIGHT-HAND-SIDE VECTOR, PLUS UPPER
BOUNDS, LOWER BOUNDS AND LINEAR COST APPROXIMATIONS FOR NON-LINEAR VARIABLE
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV9 / )
BLO(10) - SET OF LOWER BOUNDS ON NON-LINEAR VARIABLES
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV2 / )
BLT(10) - TEMPORARY STORAGE FOR BLO
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
BNDJ - BOUND ON CURRENT COLUMN
                    (GLOBAL - MAIN PROGRAM'S COMMON / MOVES / )
BOUNDS(100) - VALUES OF UPPER BOUNDS
                    (GLOBAL - MAIN PROGRAM'S COMMON / BOUNDS / )
BOUND - VALUE OF COLUMN BOUND (OR 10 ** 70)
                    ( LOCAL - MAIN PROGRAM'S BOUND )
BO(100) - RIGHT-HAND-SIDE VECTOR
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV2 / )
BUDG (10) - LIMIT ON PROCUREMENT EXPENDITURES IN EACH PERIOD
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
```

```
C - TEMP. STORAGE FOR COST DATA
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
C - TEMPORARY STORAGE LOCATION USED IN CALCULATING SAVINGS FROM RESOURCE
STORAGE
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
C - 'COLUMNS'
                    ( LOCAL - MATRIX GENERATOR'S MATFILL)
CALLER - CALLING NAME
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
CARD(8) - IMAGE OF MAP CARD
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
CFX - NO LONGER USED
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV7 / )
CHAR(5000,4) - STORAGE OF COLUMN NAMES WHICH HAVE BEEN BROKEN DOWN INTO
THEIR FOUR MEANINGFUL PARTS
                    (GLOBAL - REPORT GENERATOR'S COMMON / BASICS / )
CMAX - TEMP. COST STORAGE FOR ORDERING RESOURCES
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
CNAME - COLUMN NAME FOR WHICH RVAL IS BEING DERIVED
                    ( LOCAL - MATRIX GENERATOR'S MATFILL)
CODE(20) - STORAGE OF THE NUMBERS 1 - 20 IN TWO DIGIT ALPHANUMERIC FORM
                    (GLOBAL - REPORT GENERATOR'S COMMON / BASICS / )
COST - VALUE OF CURRENT OBJECTIVE FUNCTION
                    ( LOCAL - MAIN PROGRAM'S STATUS )
COST - TOTAL COST OF SOLUTION
                    (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
COST - COST OF THE SOLUTION RETURNED FROM THE LP
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV4 / )
COST1 - SOLUTION COST FOR LOWER BRANCH
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
COST2 - SOLUTION COST FOR UPPER BRANCH
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
COSTS (30,3) - COST OF OPERATING (1), SELLING (2) OR TRUNCATING (3) A
RESOURCE IN THE 1ST THRU 30TH YEAR OF ITS LIFE
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
CO(10) - VECTOR FOR LINEAR APPROXIMATION FOR NON-LINEAR COST FUNCTIONS
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV2 / )
CTEMP - TEMP. STORAGE FOR COLUMN NAME
                    ( LOCAL - MATRIX GENERATOR'S MATFILL)
CTOL - COST TOLERANCE FOR INFEASIBILITY
                    (GLOBAL - MAIN PROGRAM'S COMMON / TOLS / )
CT(10) - TEMPORARY STORAGE FOR CO
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
DELTA(101) - NOT USED
                    (GLOBAL - MAIN PROGRAM'S COMMON / D / )
DERTOL - DUAL ERROR TOLERANCE, NOT USED
                    (GLOBAL - MAIN PROGRAM'S COMMON / TOLS / )
DIF - DIFFERENCE BETWEEN UPPER AND LOWER BOUND FOR A VARIABLE
                    ( LOCAL - MAIN PROGRAM'S GETC )
DIF(10) - DIFFERENCES BETWEEN COST FUNCTIONS AND LINEAR APPROXIMATIONS
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
```

```
DIVOT - CANDIDATE PIVOT WHILE SEARCHING FOR BEST
                    ( LOCAL - MAIN PROGRAM'S PIVOT )
DJ(100) - VALUES OF CURRENT IN-CORE DJ'S
                    (GLOBAL - MAIN PROGRAM'S COMMON / DJS / )
DJ - DJ FOR COLUMN TO BE STORED
                    ( LOCAL - MAIN PROGRAM'S INSERT )
DJ - CURRENT COLUMN SENSITIVITY
                    ( LOCAL - MAIN PROGRAM'S CHECK )
DJTOL - DJ TOLERANCE
                    (GLOBAL - MAIN PROGRAM'S COMMON / TOLS / )
DMAX - LARGEST DJ OF STORED COLUMNS
                    (GLOBAL - MAIN PROGRAM'S COMMON / MOVES / )
DOT - DOUBLE PRECISION INNER PRODUCT OF X AND Y
                    ( LOCAL - MAIN PROGRAM'S DOT )
DOTS - SINGLE PRECISION INNER PRODUCT OF X AND Y
                    ( LOCAL - MAIN PROGRAM'S DOT )
D(15) - DJ'S OF STORED COLUMNS
                    ( LOCAL - MAIN PROGRAM'S INSERT )
DUALER - DUAL ERROR, UNUSED
                    (GLOBAL - MAIN PROGRAM'S COMMON / MOVES / )
DUM1 AND DUM2 - TEMPORARY STORAGE FOR READING UNUSED SECTIONS OF TAPE
                    ( LOCAL - MAIN PROGRAM'S INITA )
EKBL(25) - EKO VALUE ASSOCIATED WITH EACH NODE ON THE LIST
                    (GLOBAL - MAIN PROGRAM'S COMMON / CVB / )
EKO - COST ASSOCIATED WITH THE LOWER BOUNDS OF THE NODE
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
ENDER - 'END'
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
EPSI - EPSILON VALUE FROM REAL PARAMETER CARD
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
EPSI - VALUE OF NEW BASIC VARIABLE
                    ( LOCAL - MAIN PROGRAM'S PRIMAL )
ESIG - TEMPORARY STORAGE FOR EKO
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
EXIST(10,20) - NUMBER OF EACH TYPE RESOURCE AVAILABLE IN EACH YEAR
                    (GLOBAL - REPORT GENERATOR'S COMMON / OUTS / )
EXT - TIME WHEN TIME LIMIT ON PROBLEM WILL EXPIRE
                    (GLOBAL - MAIN PROGRAM'S COMMON / TMX / )
FNAME - PROBLEM TITLE (NOT USED)
                    ( LOCAL - REPORT GENERATOR'S SETUP )
FNAME - FILE NAME
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
FX1(10) - COST FUNCTION VALUES FOR LOWER BOUNDS
                    ( LOCAL - MAIN PROGRAM'S GETC )
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
FX2(10) - COST FUNCTION VALUES FOR UPPER BOUND
                    ( LOCAL - MAIN PROGRAM'S GETC )
FX2(10) - COST FUNCTION VALUES FOR SOLUTION POINT
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
GAMMA(101) - NOT USED
                    (GLOBAL - MAIN PROGRAM'S COMMON / C / )
I - DUMMY VARIABLE
                    ( LOCAL - MAIN PROGRAM'S KEYEND )
```

```
I - INPUT DISK COLUMN INDEX
                  ( LOCAL - MAIN PROGRAM'S SETBND )
IA1 - DISC FILE FOR MATRIX LESS GUB ROWS
                  (GLOBAL - MAIN PROGRAM'S COMMON / FILES / )
IA2 - DISC FILE FOR PACKED MATRIX LESS GUB ROWS
                  (GLOBAL - MAIN PROGRAM'S COMMON / FILES / )
IB - BASIC COLUMN INDEX
                  ( LOCAL - MAIN PROGRAM'S ROW )
IB - DISC INDEX OF BASIC COLUMN FOR CURRENT ROW
                  ( LOCAL - MAIN PROGRAM'S KEYCH )
IB - BOUND INDEX IN IBDS
                  ( LOCAL - MAIN PROGRAM'S BOUND )
IBAS - COUNT OF BASIS VARIABLES
                  ( LOCAL - MAIN PROGRAM'S MAPOUT )
IBASIS(101) - BASIC COLUMNS FOR NON-GUB ROWS
                  (GLOBAL - MAIN PROGRAM'S COMMON / BASIS / )
IBDS(100) - COLUMN INDICES OF BOUND COLUMNS
                  (GLOBAL - MAIN PROGRAM'S COMMON / BOUNDS / )
IBMAX - MAXIMUM NUMBER OF NODES WHICH MAY BE STORED ON BLIST
                  ( LOCAL - MAIN PROGRAM'S PRESET )
IBND - COUNT OF BOUND VARIABLES
                  ( LOCAL - MAIN PROGRAM'S MAPOUT )
IC - CURRENT COST ROW
                  (GLOBAL - MAIN PROGRAM'S COMMON / I / )
ICOST - USER'S COST ROW
                  (GLOBAL - MAIN PROGRAM'S COMMON / I / )
ICOL - TEMPORARY STORAGE FOR COLUMN INDEX
                  ( LOCAL - MAIN PROGRAM'S BBCAV2 )
ICX - NUMBER OF VARIABLES FOR WHICH COST SLOPES ARE TO BE DERIVED
                  ( LOCAL - MAIN PROGRAM'S GETC )
ID - LOCAL ROW TYPE BEING PROCESSED
                  ( LOCAL - MAIN PROGRAM'S SETUP )
ID - COLUMN NUMBER FROM MAP CARD
                  ( LOCAL - MAIN PROGRAM'S MAPIN )
ID(15) - INDICES OF STORED COLUMNS
                  ( LOCAL - MAIN PROGRAM'S INSERT )
IED - 'ENDTABLE'
                  ( LOCAL - REPORT GENERATOR'S SETUP )
IEND - LAST YEAR OF RESOURCE EXISTANCE
                  ( LOCAL - REPORT GENERATOR'S INSOLN )
IHVN (10) - POINTERS FOR INHERITED VEHICLES
                  ( LOCAL - MATRIX GENERATOR'S GENLCP )
IKEY - COUNT OF KEY VARIABLES
                  ( LOCAL - MAIN PROGRAM'S MAPOUT )
IKOST - TEMPORARY STORAGE OF USER'S COST ROW
                  ( LOCAL - MAIN PROGRAM'S SETUP )
IMAP - FILE FOR STARTING AND TERMINATING BASIS
                  (GLOBAL - MAIN PROGRAM'S COMMON / FILES / )
INDIC - INDICATES WHICH BRANCH (UPPER OR LOWER) IS BEING SOLVED
                  ( LOCAL - MAIN PROGRAM'S BBCAV2 )
IND - COLUMN NUMBER TEMPORARY STORAGE
                  ( LOCAL - REPORT GENERATOR'S INSOLN )
```

```
INH (10, 16) - NUMBER OF EACH TYPE RESOURCE INHERITED FROM EACH YEAR
                (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
INH(10,16) - NOT USED
                (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
INHYRS - NUMBER OF YEARS FROM WHICH VEHICLES ARE INHERITED
                ( LOCAL - MATRIX GENERATOR'S GENLCP )
INLL - COUNT OF NULL VARIABLES
                ( LOCAL - MAIN PROGRAM'S MAPOUT )
INON - TEMPORARY NUMBER OF NON GUB ROWS FOUND
                ( LOCAL - MAIN PROGRAM'S SETUP )
INPUT - FILE CONTAINING INPUT MATRIX
                (GLOBAL - MAIN PROGRAM'S COMMON / INPUT / )
INPUTM - NUMBER OF ROWS IN MATRIX
                (GLOBAL - MAIN PROGRAM'S COMMON / INPUT / )
INPUTN - NUMBER OF COLUMNS IN MATRIX
                (GLOBAL - MAIN PROGRAM'S COMMON / INPUT / )
INVF - INVERT FREQUENCY
                (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
INVERS - 'INVERS'
                ( LOCAL - MAIN PROGRAM'S MAPIN )
INYR - INITIAL YEAR OF PROBLEM
                (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
IOPT - USED TO FLAG UNBOUNDED SOLUTION
                (GLOBAL - MAIN PROGRAM'S COMMON / CV7 / )
IORG - ORIGIN OF BASIS INVERSE
                ( LOCAL - MAIN PROGRAM'S LP )
                ( LOCAL - MAIN PROGRAM'S ROW )
                ( LOCAL - MAIN PROGRAM'S INVERT )
IORG - ORIGIN OF PIVOT ROW IN B
                ( LOCAL - MAIN PROGRAM'S PIVOT )
IORG - ORIGIN OF A ROW IN BASIS INVERSE
                ( LOCAL - MAIN PROGRAM'S XCHECK )
                ( LOCAL - MAIN PROGRAM'S KEYCH )
                ( LOCAL - MAIN PROGRAM'S FEASCH )
IPHASE - CURRENT LP PHASE
                (GLOBAL - MAIN PROGRAM'S COMMON / I / )
IPI - CURRENT LOCATION OF PI VECTOR IN BASIS
                (GLOBAL - MAIN PROGRAM'S COMMON / I / )
IPT - 'PERIOD'
                ( LOCAL - REPORT GENERATOR'S SETUP )
IP(12) - STORAGE FOR INPUT PARAMETERS ON INTEGER PARAMETER CARD
                (GLOBAL - MAIN PROGRAM'S COMMON / CV1 / )
IROWTYP(101) - USER'S INPUT ROW TYPES
                (GLOBAL - MAIN PROGRAM'S COMMON / ROWTYP / )
IROWTP(100) - INDICATES ROW TYPE J ALL SET TO ZERO EXCEPT GENERALIZED UPPER
BOUND ROWS WHICH ARE SET TO 4
                ( LOCAL - MATRIX GENERATOR'S MATFILL)
IROW - CURRENT SELECTED ROW
                (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
IROW - ROW CALLING PARAMETER, ROW OR ZERO
                ( LOCAL - MAIN PROGRAM'S ROW )
IROW - ROW TO WHICH KEY COLUMN IS SHIFTED WHEN MADE BASIC
                ( LOCAL - MAIN PROGRAM'S KEYCH )
IROW - PIVOT ROW
                ( LOCAL - MAIN PROGRAM'S PIVOT )
```

```
ISTART - FIRST YEAR OF RESOURCE EXISTANCE
               ( LOCAL - REPORT GENERATOR'S INSOLN )
ITABLE - TABLE NAME
               ( LOCAL - MATRIX GENERATOR'S GENLCP )
               ( LOCAL - REPORT GENERATOR'S SETUP )
ITEMP - TEMP. STORAGE FOR FIRST LETTER OF RNAME
               ( LOCAL - MATRIX GENERATOR'S MATFILL)
ITNINV - ITERATION OF NEXT INVERT
               (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
ITRN - ITERATION COUNT
               (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
ITYPE - ROW TYPE
               ( LOCAL - MAIN PROGRAM'S INVERT )
ITYPE - TYPE OF STEP, 1-UNBOUNDED, 2-COLUMN TO ZERO, 3-COLUMN   BOUND
               ( LOCAL - MAIN PROGRAM'S ROW )
               ( LOCAL - MAIN PROGRAM'S PRIMAL )
IVT - 'VEHICLE'
               ( LOCAL - REPORT GENERATOR'S SETUP )
IX(100) - INDICES OF SOLUTION COLUMNS
               (GLOBAL - MAIN PROGRAM'S COMMON / IXX / )
IX(110) - COLUMNS IN BASIC SOLUTION
               (GLOBAL - MAIN PROGRAM'S COMMON / CV4 / )
IXZ(110) - COLUMNS IN BEST SOLUTION
               (GLOBAL - MAIN PROGRAM'S COMMON / CV4 / )
IYR(10) - INITIAL YEAR OF EACH PERIOD
               (GLOBAL - REPORT GENERATOR'S COMMON / BASICS / )
J - ABSOLUTE VALUE OF I
               ( LOCAL - MAIN PROGRAM'S SETBND )
J - INPUT DISC COLUMN INDEX
               ( LOCAL - MAIN PROGRAM'S BOUND )
J1 - FIRST COLUMN IN COLUMN PRINTOUT
               ( LOCAL - MAIN PROGRAM'S XCHECK )
J2 - LAST COLUMN IN COLUMN PRINTOUT
               ( LOCAL - MAIN PROGRAM'S XCHECK )
JAJ - DISK INDEX OF AN IN-CORE COLUMN
               ( LOCAL - MAIN PROGRAM'S XCHECK )
JAJ - POTENTIAL COLUMN'S IN-CORE POSITION
               ( LOCAL - MAIN PROGRAM'S KEYEND )
JAK(101) - DUMMY STORAGE AREA
               (GLOBAL - MAIN PROGRAM'S COMMON / CORE / )
JAREJ(101) - SET TO L WHEN CORRESPONDING IN-CORE COLUMN REJECTED
               (GLOBAL - MAIN PROGRAM'S COMMON / CORE / )
JA(101) - IN-CORE COLUMN DISC INDICES
               (GLOBAL - MAIN PROGRAM'S COMMON / CORE / )
JBMAX - MAXIMUM NUMBER OF WORDS OF INFORMATION WHICH MAY BE STORED FOR
EACH NODE IN BLIST
               ( LOCAL - MAIN PROGRAM'S PRESET )
JCOL - CURRENT SELECTED COLUMN
               (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
JCOL - USER'S COLUMN INDEX OF COLUMN PROCESSED
               ( LOCAL - MAIN PROGRAM'S MAPOUT )
JCOL - CORE POSITION OF SELECTED COLUMN
               ( LOCAL - MAIN PROGRAM'S COLUMN )
               ( LOCAL - MAIN PROGRAM'S IO )
               ( LOCAL - MAIN PROGRAM'S ROW )
```

```
JCOL - COLUMN TO BE MOVED
                    ( LOCAL - MAIN PROGRAM'S KEYCH )
JCOLPK - GUB PACKET OF COLUMN BEING MOVED FROM KEY
                    ( LOCAL - MAIN PROGRAM'S KEYCH )
JCOUNT - COUNT OF COLUMNS PROCESSED
                    ( LOCAL - MAIN PROGRAM'S CHECK )
JEND - ORIGIN OF VACANT WORK SPACE IN AJ
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
JEBCH - NUMBER OF COLUMNS CHECKED IN CURRENT BATCH
                    ( LOCAL - MAIN PROGRAM'S CHECK )
JKEY - CORE POSITION OF KEY FOR JCOL (IF GUB ROW)
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
JKEY - CANDIDATE KEY COLUMN
                    ( LOCAL - MAIN PROGRAM'S KEYCH )
JNCORE - NUMBER OF COLUMNS IN CORE
                    (GLOBAL - MAIN PROGRAM'S COMMON / LIMS / )
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
JNSCAN - COLUMNS IN CORE + 1000* NUMBER OF REWINDS OF FILE IA1
                    ( LOCAL - MAIN PROGRAM'S STATUS )
JNT - INDEX OF COLUMNS READ
                    ( LOCAL - MAIN PROGRAM'S IO )
JNT - CURRENT COLUMN INDEX
                    ( LOCAL - MAIN PROGRAM'S CHECK )
                    ( LOCAL - MAIN PROGRAM'S INVERT )
JNT - LAST COLUMN READ FROM DISC (IF MNTRY = 0)
                    ( LOCAL - MAIN PROGRAM'S STATUS )
JNTO - INDEX OF LAST COLUMN READ FROM DISC
                    ( LOCAL - MAIN PROGRAM'S STATUS )
JORG - CORE ORIGIN OF SELECTED COLUMN
                    ( LOCAL - MAIN PROGRAM'S ROW )
JORG - ORIGIN OF VACANCY FOR COLUMN IN AJ
                    ( LOCAL - MAIN PROGRAM'S INSERT )
JORG - ORIGIN IN AJ TO WHICH COLUMN IS READ
                    ( LOCAL - MAIN PROGRAM'S CHECK )
                    ( LOCAL - MAIN PROGRAM'S INVERT )
JORG - ORIGIN OF A ROW IN B
                    ( LOCAL - MAIN PROGRAM'S KEYCH )
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
                    ( LOCAL - MAIN PROGRAM'S PIVOT )
JORG - ORIGIN OF A COLUMN IN AJ
                    ( LOCAL - MAIN PROGRAM'S XCHECK )
JOUT - REJECTED COLUMN INDEX
                    (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
JOUT - COLUMN TO BE REJECTED
                    ( LOCAL - MAIN PROGRAM'S ROW )
JOUTPK - GUB PACKET OF COLUMN REJECTED
                    ( LOCAL - MAIN PROGRAM'S PRIMAL )
JP - BASIC COLUMN FOR A ROW
                    ( LOCAL - MAIN PROGRAM'S PIVOT )
JPKT - GUB PACKET OF NEW COLUMN
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
                    ( LOCAL - MAIN PROGRAM'S ROW )
```

```
JPKT - GUB PACKET OF CURRENT COLUMN
                   ( LOCAL - MAIN PROGRAM'S FEASCH )
                   ( LOCAL - MAIN PROGRAM'S KEYFND )
JPKTO - CURRENT STORED GUB KEY PACKET
                   ( LOCAL - MAIN PROGRAM'S COLUMN )
JPOS - SELECTED COLUMN INDEX
                   (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
JPOS - DISC INDEX OF COLUMN SELECTED
                   ( LOCAL - MAIN PROGRAM'S ROW )
JPOS - DISC INDEX OF COLUMN TO BE STORED
                   ( LOCAL - MAIN PROGRAM'S INSERT )
JPOSPK - GUB PACKET OF COLUMN ENTERING
                   ( LOCAL - MAIN PROGRAM'S PRIMAL )
JPOSR- DISC INDEX OF COLUMN TO BE REJECTED
                   ( LOCAL - MAIN PROGRAM'S INSERT )
JREJ - ORIGIN OF REJECTED COLUMN IN AJ
                   ( LOCAL - MAIN PROGRAM'S INSERT )
JRHS - USER'S INPUT RHS
                   (GLOBAL - MAIN PROGRAM'S COMMON / I / )
JSUB(10) - POINTERS FOR VEHICLE SUBSCRIPTS
                   ( LOCAL - MATRIX GENERATOR'S YINTERP )
JTYPE - VARIABLE TYPE
                   ( LOCAL - MAIN PROGRAM'S INVERT )
JTYPE - COLUMN TYPE
                   ( LOCAL - MAIN PROGRAM'S KEYFND )
                   ( LOCAL - MAIN PROGRAM'S CHECK )
                   ( LOCAL - MAIN PROGRAM'S COLUMN )
K - NEW STATE
                   ( LOCAL - MAIN PROGRAM'S SETBND )
K1 - NOT USED
                   (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
K2 - NOT USED
                   (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
K3 - OUTPUT CONTROL PARAMETER
                   (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
K4 - XCHECK CONTROL PARAMETER
                   (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
K5 - MAXIMUM LP ITERATIONS BEFORE MAPOUT
                   (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
KEE - 'KEY'
                   ( LOCAL - MAIN PROGRAM'S MAPIN )
KEY - INDEX OF KEY COLUMN TO BE LOCATED
                   ( LOCAL - MAIN PROGRAM'S IO )
                   ( LOCAL - MAIN PROGRAM'S KEYFND )
KEY - SWITCH TO RETURN KEY PROCESSING TO KEY LOOP
                   ( LOCAL - MAIN PROGRAM'S FEASCH )
KEYS(101)-STORAGE OF GUB KEY COLUMNS
                   (GLOBAL - MAIN PROGRAM'S COMMON / BASIS / )
KEYFND - POSITION OF KEY FOUND
                   ( LOCAL - MAIN PROGRAM'S KEYFND )
KOL - COLUMN TO BE LOCATED ON EITHER FILE, OR PACKET NUMBER OF DESIRED KEY
                   ( LOCAL - MAIN PROGRAM'S IO )
KOL1 - LAST COLUMN READ, ON FILE IA1
                   ( LOCAL - MAIN PROGRAM'S IO )
```

```
KOL2 - LAST COLUMN READ ON FILE IA2
                    ( LOCAL - MAIN PROGRAM'S IO )
KORG - ORIGIN IN AJ TO WHICH KEY COLUMN IS READ
                    ( LOCAL - MAIN PROGRAM'S INVERT )
KORG - ORIGIN OF KEY COLUMN FOR PACKET JPKT
                    ( LOCAL - MAIN PROGRAM'S ROW )
KORG - ORIGIN OF KEY COLUMN IN AJ
                    ( LOCAL - MAIN PROGRAM'S CHECK )
                    ( LOCAL - MAIN PROGRAM'S COLUMN )
L - NUMBER OF GUB ROWS
                    (GLOBAL - MAIN PROGRAM'S COMMON / I / )
LAST - LAST YEAR OF PROBLEM
                    (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
LEND - LENGTH OF PERIOD UNDER CONSIDERATION
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
LIFER - TEMP. STORAGE FOR REMAINING USEFUL LIFE OF A VEHICLE
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
LSTMAX - MAXIMUM LENGTH WHICH THE BRANCHING LIST HAS ACHIEVED
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV5 / )
LSTFRE(25) - GIVES LOCATIONS OF STORAGE AREAS ON THE BRANCHING LIST WHICH
ARE VACANT
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
LY - LAST YEAR OF A PROBLEM
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
LYR(10) - LAST YEAR OF EACH PERIOD
                    (GLOBAL - REPORT GENERATOR'S COMMON / BASICS / )
M - NUMBER OF ACTIVE INTERNAL ROWS
                    (GLOBAL - MAIN PROGRAM'S COMMON / I / )
M - NUMBER OF ROWS IN MATRIX
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
M - VECTOR DIMENSION
                    ( LOCAL - MAIN PROGRAM'S DOT )
MAPBND(100) - BOUND COLUMN INDICES
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
MAPBAS(100) - BASIC COLUMN INDICES
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
MAPKEY(1000) - KEY COLUMN INDICES
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
MAPNLL(10) - NULL COLUMN INDICES
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
MAXL - TEMP. STORAGE FOR VEHICLE LIFE
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
MAXTRY - MAXIMUM NUMBER OF IN-CORE ITERATIONS
                    (GLOBAL - MAIN PROGRAM'S COMMON / LIMS / )
MC - LAST LOGICAL COLUMN
                    (GLOBAL - MAIN PROGRAM'S COMMON / I / )
MCOL - NUMBER OF COLUMNS IN MATRIX
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
MM - NUMBER OF ELEMENTS IN BASIS INVERSE
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
MNC - THE NEGATIVE OF NCF
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
```

```
MNTRY - NUMBER OF IN-CORE ITERATIONS
                    ( LOCAL - MAIN PROGRAM'S STATUS )
MNX - THE NEGATIVE OF N
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
MP1 - M PLUS 1
                    ( LOCAL - MAIN PROGRAM'S MAPOUT )
MPK - ROW OF COLUMN WHICH WAS KEY
                    ( LOCAL - MAIN PROGRAM'S KEYCH )
MPLUS - NUMBER OF ROWS IN THE MATRIX INCLUDING THE COST ROW (M+1)
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
MPL - M PLUS L
                    (GLOBAL - MAIN PROGRAM'S COMMON / I / )
MROWS - USER'S NUMBER OF ROWS
                    ( LOCAL - MAIN PROGRAM'S LP )
N - NUMBER OF COLUMNS IN MATRIX
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
N - NUMBER OF COLUMNS CURRENTLY SAVED
                    ( LOCAL - MAIN PROGRAM'S INSERT )
NAAM - NOT USED
                    ( LOCAL - MAIN PROGRAM'S IO )
NAMEN (10) - POINTERS FOR NUMBERING RESOURCES
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
NAMEN(10) - NOT USED
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
NAMES(5) - COLUMN INDICES FROM MAP CARD
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
NAMES (10) - TEMPORARY POINTERS
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
NAME(600) - STATE OF EACH VARIABLE OR COLUMN
                    (GLOBAL - MAIN PROGRAM'S COMMON / NAMES / )
NAME - COLUMN NAME, OR POSITION IN CORE TO WHICH COLUMN IS READ
                    ( LOCAL - MAIN PROGRAM'S IO )
NB - NUMBER OF BASIC VARIABLES IN A PACKET
                    ( LOCAL - MAIN PROGRAM'S FEASCH )
NBCH - NUMBER OF COLUMNS IN BATCH
                    ( LOCAL - MAIN PROGRAM'S CHECK )
NBDS - NUMBER OF BOUNDS
                    (GLOBAL - MAIN PROGRAM'S COMMON / BOUNDS / )
NBVPKT - NUMBER OF BASIS VARIABLE IN SELECTED GUB ROW
                    ( LOCAL - MAIN PROGRAM'S PRIMAL )
NC - NUMBER OF BASIS CARDS TO BE READ FROM INPUT
                    ( LOCAL - MAIN PROGRAM'S READIN )
NCF - NUMBER OF NON-LINEAR VARIABLES
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
NCF1 - NCF
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
NCF4 - NORA + 3.* NCF
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
NCHGS - USER'S NUMBER OF BOUND COLUMNS
                    ( LOCAL - MAIN PROGRAM'S LP )
NCOLS - NUMBER OF COLUMNS READ FROM DISC FILE IA1
                    ( LOCAL - MAIN PROGRAM'S STATUS )
```

```
NCOLS - USER'S NUMBER OF COLUMNS
                ( LOCAL - MAIN PROGRAM'S LP )
NCORE - NUMBER OF COLUMNS IN CORE
                ( LOCAL - MAIN PROGRAM'S COLUMN )
NCRMAX - MAXIMUM NUMBER OF COLUMNS WHICH FIT IN CORE
                (GLOBAL - MAIN PROGRAM'S COMMON / LIMS / )
NDEGLM - MAXIMUM REJECTION DUE TO DEGENERACY
                ( LOCAL - MAIN PROGRAM'S PRIMAL )
NDEG - NUMBER OF DEGENERACY REJECTIONS
                ( LOCAL - MAIN PROGRAM'S PRIMAL )
NDJS - NUMBER OF NEGATIVE DJ'S
                (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
NDJST - NUMBER OF NEGATIVE DJ'S FROM DISK READ
                ( LOCAL - MAIN PROGRAM'S COLUMN )
NDX(10) - INDICES OF NON-LINEAR VARIABLES
                ( LOCAL - MAIN PROGRAM'S NXBRN )
NEM1 - NUMBER OF ELEMENTS IN AN ARRAY MINUS ONE
                ( LOCAL - MAIN PROGRAM'S GETASQ )
NEWXZ - FLAGS WHEN NEW BEST SOLUTION FOUND AND SHOULD BE OUTPUT
                (GLOBAL - MAIN PROGRAM'S COMMON / CV7 / )
NEWROW - ROW FOR COLUMN CHANGING FROM KEY TO BASIC
                ( LOCAL - MAIN PROGRAM'S PRIMAL )
NF1 - SIGNIFIES FEASIBLE SOLUTION WHEN SET EQUAL TO 1
                (GLOBAL - MAIN PROGRAM'S COMMON / CV7 / )
NFBCH - NUMBER OF COLUMNS RETAINED FROM BATCH
                ( LOCAL - MAIN PROGRAM'S CHECK )
NFREE - NUMBER OF GAPS (EMPTY LOCATION BETWEEN TWO FILLED LOCATIONS) IN
THE BLIST
                ( LOCAL - MAIN PROGRAM'S BBCAV2 )
NFX - THE NEGATIVE OF NCF
                ( LOCAL - MAIN PROGRAM'S NXBRN )
NINHP - NUMBER OF INHERITED PERIODS
                ( LOCAL - MATRIX GENERATOR'S GENLCP )
NIV - NUMBER OF INHERITED VEHICLE TYPES
                ( LOCAL - MATRIX GENERATOR'S GENLCP )
NJOUT - NAME CODE OF COLUMN TO BE REJECTED
                ( LOCAL - MAIN PROGRAM'S STATUS )
NL (10) - TEMP. STORAGE USED IN FORMATTING OUTPUT
                ( LOCAL - MATRIX GENERATOR'S GENLCP )
NMIN - INDEX OF THE LOWEST BOUND ON THE BLIST, OR N-1, DEPENDING ON WHERE
IT IS USED
                ( LOCAL - MAIN PROGRAM'S BBCAV2 )
NN (10) TEMP. STORAGE USED IN FORMATTING OUTPUT
                ( LOCAL - MATRIX GENERATOR'S GENLCP )
NOBOL - NUMBER OF NODES ON LIST
                (GLOBAL - MAIN PROGRAM'S COMMON / CV8 / )
NOL - INDEX FOR STORAGE ON BLIST
                ( LOCAL - MAIN PROGRAM'S BBCAV2 )
NOP - NODE NUMBER
                (GLOBAL - MAIN PROGRAM'S COMMON / CV7 / )
NOPS - NODES SOLVED
                (GLOBAL - MAIN PROGRAM'S COMMON / CV7 / )
```

```
NORA - NUMBER OF ROWS IN THE MATRIX INCLUDING THE OBJECTIVE FUNCTION
                 ( LOCAL - MAIN PROGRAM'S BRCAV2 )
                 ( LOCAL - MAIN PROGRAM'S PRESET )
NOTE(4) - 40 CHARACTER COMMENT
                 ( LOCAL - MAIN PROGRAM'S STATUS )
NP' - NUMBER OF SUBPERIODS
                 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
NPBCH - NUMBER OF COLUMNS TO BE SAVED FROM BATCH
                 ( LOCAL - MAIN PROGRAM'S INSERT )
NPERYR (10, 3) - FIRST AND LAST YEAR OF PERIOD AND NUMBER OF TASKS IN PERIOD
                 (GLOBAL - MATRIX GENERATOR'S COMMON / PRDSIG / )
NPHASE - STORES LP PHASE CODE
                 (GLOBAL - MAIN PROGRAM'S COMMON / CV/ / )
NPIF - NUMBER OF PRIMAL INFEASIBILITIES
                 (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
NPP - NUMBER OF PERIODS
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
NPTASK (10, 9) - ID NUMBER OF EACH TASK IN PERIOD
                 (GLOBAL - MATRIX GENERATOR'S COMMON / PRDSIG / )
NPT - NUMBER OF PERIOD TABLES READ
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
                 ( LOCAL - REPORT GENERATOR'S SETUP )
NRD - NUMBER OF VEHICLES HAVING R AND D
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
NREJ - NUMBER OF REJECTED IN-CORE COLUMNS
                 (GLOBAL - MAIN PROGRAM'S COMMON / STATE / )
NROW - NUMBER OF ROWS IN MATRIX
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
NV - NUMBER OF RESOURCE TYPES
                 (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
NSCAN - NUMBER OF DISC READS
                 (GLOBAL - MAIN PROGRAM'S COMMON / LIMS / )
NT - NUMBER OF TASKS
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
NT - TOTAL NUMBER OF COLUMNS (MC+INPUTN)
                 (GLOBAL - MAIN PROGRAM'S COMMON / 1 / )
NT - NUMBER OF TASKS (NOT USED)
                 ( LOCAL - REPORT GENERATOR'S SETUP )
NTR - NUMBER OF TASK TABLES READ
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
NTRY - NUMBER OF IN-CORE ITERATIONS
                 (GLOBAL - MAIN PROGRAM'S COMMON , LIMS , )
NTSK (9) - NUMBER OF ALTERNATIVES IN TASK
                 (GLOBAL - MATRIX GENERATOR'S COMMON / TSKSIG / )
NULL - 'NULL'
                 ( LOCAL - MAIN PROGRAM'S MAPIN )
NULL - COLUMN STATE
                 ( LOCAL - MAIN PROGRAM'S COLUMN )
NV - NUMBER OF VEHICLE TYPES
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
NVEHU (10) - INDICATES IF VEHICLE USED IN PERIOD
                 ( LOCAL - MATRIX GENERATOR'S GENLCP )
```

```
NVR - NUMBER OF VEHICLE TABLES READ
              ( LOCAL - MATRIX GENERATOR'S GENLCP )
              ( LOCAL - REPORT GENERATOR'S SETUP )
NWAJ - STORAGE DIMENSION OF THE ARRAY AJ
              ( LOCAL - MAIN PROGRAM'S LP )
NXBX - INDEX OF BRANCHING VARIABLE
              (GLOBAL - MAIN PROGRAM'S COMMON / CVB / )
NXBL(25) - INDEX OF BRANCHING VARIABLE FOR EACH NODE
              (GLOBAL - MAIN PROGRAM'S COMMON / CV9 / )
NXB - TEMPORARY STORAGE FOR NEXT BRANCHING VARIABLE
              ( LOCAL - MAIN PROGRAM'S BBCAV2 )
NYR - TEMP. STORAGE FOR LAST YEAR OF PERIOD
              ( LOCAL - MATRIX GENERATOR'S GENLCP )
OANDM(20) - OPERATING COST FOR EACH YEAR
              (GLOBAL - REPORT GENERATOR'S COMMON / OUTS / )
ONF - '01'
              ( LOCAL - REPORT GENERATOR'S CINFO )
ONE - 1.0
              ( LOCAL - MATRIX GENERATOR'S GENLCP )
ONEM - -1.0
              ( LOCAL - MATRIX GENERATOR'S GENLCP )
PACK(100) - TEMPORARY STORAGE OF PACKED COLUMN
              ( LOCAL - MAIN PROGRAM'S IO )
PER(10) - POINTERS FOR TWO DIGIT, ALPHANUMERIC CODE FOR PERIODS
              (GLOBAL - REPORT GENERATOR'S COMMON / BASICS / )
PERIOD - 'PERIOD'
              ( LOCAL - REPORT GENERATOR'S CINFO )
              ( LOCAL - REPORT GENERATOR'S PINFO )
PH1 AND PH2 - TEMPORARY STORAGE OF VALUES FROM GETPHI
              ( LOCAL - MAIN PROGRAM'S BBCAV2 )
PHIT - COST OF A NON-LINEAR SOLUTION
              (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
PIKEY - DJ VALUE FOR CURRENT KEY JPKT
              ( LOCAL - MAIN PROGRAM'S COLUMN )
PIKEY - DJ FOR CURRENT KEY AT KORG
              ( LOCAL - MAIN PROGRAM'S CHECK )
PIVTOL - PIVOT TOLERANCE
              (GLOBAL - MAIN PROGRAM'S COMMON / TOLS / )
PIV - PIVOT USED
              ( LOCAL - MAIN PROGRAM'S PIVOT )
PKT1 - TEMPORARY COUNT OF GUB ROW PACKET COLUMNS
              ( LOCAL - MAIN PROGRAM'S SETUP )
PKT - ACTUAL GUB ROW COLUMN BEING PROCESSED
              ( LOCAL - MAIN PROGRAM'S SETUP )
PKT - STORAGE OF COLUMN PACKET
              ( LOCAL - MAIN PROGRAM'S MAPIN )
PKT - GUB PACKET NUMBER OF COLUMN BEING PROCESSED
              ( LOCAL - MAIN PROGRAM'S INVERT )
PKT - PACKET OF NEW COLUMN, JNT
              ( LOCAL - MAIN PROGRAM'S CHECK )
PKT - GUB PACKET OF DESIRED KEY
              ( LOCAL - MAIN PROGRAM'S KEYEND )
PKTO - GUB PACKET NUMBER OF COLUMN IN AJ(KORG)
PKTO - PACKET OF CURRENT KEY
              ( LOCAL - MAIN PROGRAM'S CHECK )
              ( LOCAL - MAIN PROGRAM'S INVERT )
```

PMIN - VALUE OF LOWEST BOUND ON BLIST
(LOCAL - MAIN PROGRAM'S BBCAV2)

PERTOL - PRIMAL ERROR TOLERANCE
(GLOBAL - MAIN PROGRAM'S COMMON / TOLS /)

PRMLER - PRIMAL ERROR, UNUSED
(GLOBAL - MAIN PROGRAM'S COMMON / MOVES /)

PROC(20) - PROCUREMENT FUNDS SPENT DURING EACH PERIOD
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

PROT(20) - PROCUREMENT FUNDS AVAILABLE DURING EACH PERIOD
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

PSIGL(25) - LOWER BOUND ASSOCIATED WITH EACH NODE ON LIST
(GLOBAL - MAIN PROGRAM'S COMMON / CV9 /)

PTASK (10, 9) - MULTIPLICATIVE FACTOR FOR ALL VALUES IN ASSOCIATED TASK FOR EACH PERIOD
(GLOBAL - MATRIX GENERATOR'S COMMON / PRDSIG /)

PURCH(10,20) - NUMBER OF EACH TYPE RESOURCE PURCHASED IN EACH YEAR
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

R - RATE OF INCREASE IN OPERATING COST
(LOCAL - MATRIX GENERATOR'S YRCOST)

R - 'RHS'
(LOCAL - MATRIX GENERATOR'S MATFILL)

R1 - PORTION OF OPERATING COST REFUNDED FOR MOTH BALLING RESOURCE
(LOCAL - MATRIX GENERATOR'S YRCOST)

RDTOT - TOTAL R AND D EXPENDITURES
(GLOBAL - REPORT GENERATOR'S COMMON / PARAMS /)

RHS(100) - STORES USER'S CURRENT RIGHT HAND SIDE
(GLOBAL - MAIN PROGRAM'S COMMON / RHS /)

RNAME(120) - ROW NAMES
(LOCAL - MATRIX GENERATOR'S MATFILL)

ROWS - 'ROWS'
(LOCAL - MAIN PROGRAM'S MAPIN)

RP(12) - STORAGE FOR REAL PARAMETERS, FIRST FOUR LOCATIONS ARE FOR INPUT FROM REAL PARAMETER CARD, REST ARE TEMPORARY STORAGE
(GLOBAL - MAIN PROGRAM'S COMMON / CV1 /)

RTEMP - TEMP. STORAGE FOR ROW NAMES
(LOCAL - MATRIX GENERATOR'S MATFILL)

RVAL(100) - VECTOR OF VALUES IN EACH ROW FOR A SPECIFIC COLUMN
(LOCAL - MATRIX GENERATOR'S MATFILL)

S - 'S'
(LOCAL - REPORT GENERATOR'S INSOLN)

SALE(20) - SALVAGE OR TRUNCATION VALUE FOR EACH YEAR
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

SALV(10,20) - NUMBER OF EACH TYPE RESOURCE DISPOSED OF AT END OF EACH YEAR
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

SAVE(20) - SAVINGS FROM RESOURCE STORAGE FOR EACH YEAR
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

SECS - ACTUAL CPU CLOCK TIME
(LOCAL - MAIN PROGRAM'S TIMEC)

SIGMA(100,4) - STORES INFORMATION WHICH DEFINES THE CURRENT NODE
(GLOBAL - MAIN PROGRAM'S COMMON / CV5 /)

STEP - STEP TO CURRENT ROW
(LOCAL - MAIN PROGRAM'S ROW)

STOR(10,20) - NUMBER OF EACH TYPE RESOURCE STORED IN EACH YEAR
(GLOBAL - REPORT GENERATOR'S COMMON / OUTS /)

```
SUM - VALUE OF VARIABLE BEFORE FEASIBILITY ADJUSTMENT
                ( LOCAL - MAIN PROGRAM'S FEASCH )
SUM - TEMPORARY STORAGE
                ( LOCAL - MAIN PROGRAM'S KEYCH )
SUM - DOUBLE PRESISION ACCUMULATOR
                ( LOCAL - MAIN PROGRAM'S DOT )
SUM - TOTAL COST FOR A PERIOD
                ( LOCAL - REPORT GENERATOR'S CINFO )
SUMIE - SUM OF INFEASIBILITIES
                ( LOCAL - MAIN PROGRAM'S FEASCH )
SUMT - TOTAL COST FOR ALL PERIODS
                ( LOCAL - REPORT GENERATOR'S CINFO )
SY - START YEAR OF PROBLEM
                ( LOCAL - MATRIX GENERATOR'S GENLCP )
T(100,10) - STORAGE FOR COLUMNS OF MATRIX ASSOCIATED WITH NON-LINEAR
VARIABLES
                (GLOBAL - MAIN PROGRAM'S COMMON / CV2 / )
TCOST - TEMP STORAGE FOR TOTAL PROCUREMENT
                ( LOCAL - REPORT GENERATOR'S CINFO )
TEMP - LOCATION USED WHILE SWAPPING CONTENTS OF TWO LOCATIONS IN AN ARRAY
                ( LOCAL - MAIN PROGRAM'S GETASQ )
TEMP1 - TEMPORARY STORAGE LOCATIONS FOR ALPHANUMERIC OUTPUT
                ( LOCAL - REPORT GENERATOR'S PINFO )
TEMP2 - TEMPORARY STORAGE LOCATIONS FOR ALPHANUMERIC OUTPUT
                ( LOCAL - REPORT GENERATOR'S PINFO )
TEMP(4) - TEMP STORAGE FOR COLUMN NAMES
                ( LOCAL - REPORT GENERATOR'S SETUP )
THETA - STEP CHOSEN BY ROW, ADJUSTED IN PRIMAL
                (GLOBAL - MAIN PROGRAM'S COMMON / MOVES / )
THETA - BEST FEASIBLE STEP
                ( LOCAL - MAIN PROGRAM'S ROW )
TITLE(4) - ALPHANUMERIC TITLE OF PROBLEM
                ( LOCAL - MAIN PROGRAM'S BBCAV2 )
                (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
TMAX - MAXIMUM TIME BEFORE MAPOUT
                (GLOBAL - MAIN PROGRAM'S COMMON / PARAMS / )
TMO - TIME SET WAS CALLED
                (GLOBAL - MAIN PROGRAM'S COMMON / TMX / )
TMP(10) - TEMPORARY STORAGE
                (GLOBAL - MAIN PROGRAM'S COMMON / CV1 / )
TOT - NUMBER OF SUBPERIODS PLUS 1
                (GLOBAL - REPORT GENERATOR'S COMMON / PARAMS / )
TOTAL - 'TOTAL'
                ( LOCAL - REPORT GENERATOR'S CINFO )
                ( LOCAL - REPORT GENERATOR'S PINFO )
TPROC - CORRECTION FACTOR FOR PROCUREMENT
                ( LOCAL - REPORT GENERATOR'S CINFO )
TSIG - TEMPORARY STORAGE ASSOCIATED WITH EKO
                (GLOBAL - MAIN PROGRAM'S COMMON / CV5 / )
TSTO(130) - TEMPORARY STORAGE
                ( LOCAL - MAIN PROGRAM'S BBCAV2 )
TYPE1 - FIRST WORD ON MAP CARD
                ( LOCAL - MAIN PROGRAM'S MAPIN )
```

```
TYPE2 - SECOND WORD ON MAP CARD
                    ( LOCAL - MAIN PROGRAM'S MAPIN )
U (7, 288, 9) - ARRAY OF TASK ALTERNATIVES
                    (GLOBAL - MATRIX GENERATOR'S COMMON / TSKSTG / )
UB (10) - CALCULATED UPPER BOUNDS ON RESOURCES
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
UBK - UPPER BOUND ON BRANCHING VARIABLE
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
UBK2 - DIFFERENCE BETWEEN UPPER BOUND AND VALUE FOR BRANCHING VARIABLE
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
ULO(10) - SET OF UPPER BOUNDS ON NON-LINEAR VARIABLES
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV2 / )
ULT(10) - TEMPORARY STORAGE FOR ULO
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
UMAX - TEMP. STORAGE FOR GREATEST QUANTITY OF A SPECIFIC VEHICLE WHICH MIGHT
BE USED IN A TASK
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
US - TEMPORARY STORAGE FOR USP
                    ( LOCAL - MAIN PROGRAM'S BBCAV2 )
USM = UZ/(1-E)
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
USP = UZ/(1+E)
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
UZ - COST OF BEST NON-LINEAR SOLUTION
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV3 / )
VAL - COLUMN VALUE TEMPORARY STORAGE
                    ( LOCAL - REPORT GENERATOR'S INSOLN )
VAL - TEMP. STORAGE FOR VALUE OF SPECIFIC ROW AND COLUMN
                    ( LOCAL - MATRIX GENERATOR'S MATFILL)
VCOST(10,5) - THE FIVE COSTS ASSOCIATED WITH EACH RESOURCE ARE STORED IN
THIS ARRAY - IN ORDER, THEY ARE SALVAGE AND TRUNCATION, OPERATING, R AND D,
RETENTION RATE, AND PROCUREMENT.
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
VLIFE (10) - MAXIMUM LIFE OF RESOURCE (VEHICLE)
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
VMIN  - TEMP. STORAGE FOR MINIMUM QUANTITY OF VEHICLES WHICH CAN BE USED FOR
TASK
                    ( LOCAL - MATRIX GENERATOR'S YINTERP )
VNAME(10) - STORES RESOURCE NAMES
                    (GLOBAL - MATRIX GENERATOR'S COMMON / VECSTG / )
                    (GLOBAL - REPORT GENERATOR'S COMMON / VECSTG / )
W - 'W'
                    ( LOCAL - REPORT GENERATOR'S INSOLN )
X - INPUT VECTOR
                    ( LOCAL - MAIN PROGRAM'S DOT )
X - 'X'
                    ( LOCAL - REPORT GENERATOR'S INSOLN )
X - ELAPSED CPU SECONDS
                    ( LOCAL - MAIN PROGRAM'S STATUS )
XCON(10) - STORES VALUES FOUND IN X WHICH ARE ASSOCIATED WITH THE
NON-LINEAR VARIABLES
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV4 / )
```

```
XK - VALUE OF BRANCHING VARIABLE
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV8 / )
XNXBL(25) - VALUE OF BRANCHING VARIABLE FOR EACH NODE
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV9 / )
XT(10) - SOLUTION VALUES FOR NON-LINEAR VARIABLE
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
XX- ELAPSED TIME ON PROBLEM
                    ( LOCAL - MAIN PROGRAM'S TIMEC )
X(100) - VALUES OF SOLUTION COLUMNS
                    (GLOBAL - MAIN PROGRAM'S COMMON / XX / )
X(110) - VALUES ASSOCIATED WITH COLUMNS IN IX
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV4 / )
XZ(110) - VALUES ASSOCIATED WITH COLUMNS IN IXZ
                    (GLOBAL - MAIN PROGRAM'S COMMON / CV4 / )
Y - INPUT VECTOR
                    ( LOCAL - MAIN PROGRAM'S DOT )
YAVL (10) - YEAR RESOURCE FIRST AVAILABLE
                    (GLOBAL - MATRIX GENERATOR'S COMMON / ALTSTG / )
YEARS (21) - STORES INHERITED YEARS
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
YRINT (20) - SCALE FACTOR FOR ALL TASKS IN PERIOD
                    ( LOCAL - MATRIX GENERATOR'S GENLCP )
YT(10) - DIFFERENCES BETWEEN SOLUTION POINT AND LOWER BOUNDS
                    ( LOCAL - MAIN PROGRAM'S NXBRN )
Z - PARAMETER USED TO PACK INDEX OF COEFFICIENT
                    ( LOCAL - MAIN PROGRAM'S IO )
G9H8 8 7==''977 H9'8Z9IG9= 97='9H
                    (GLOBAL - MAIN PROGRAM'S COMMON / TOLS / )
ZS - PARAMETER USED TO PACK COEFFICIENTS
                    ( LOCAL - MAIN PROGRAM'S IO )
```

REFERENCES FOR VOLUME I

1. J. C. Hetrick, "Mathematical Models in Capital Budgeting," Chapter 7. New Decision Making Tools for Managers; edited by E. C. Bursk and J. F. Chapman, (Cambridge, Harvard University Press, 1963) p. 186.

2. J. E. Falk and R. M. Soland, "An Algorithm for Separable Non-Convex Programming Problems," Management Science, Vol. 15, No. 9, May 1969.

3. E. M. L., Beale, "Advanced Algorithmic Features for General Mathematical Programming Systems," Integer and Nonlinear Programming, edited by J. Abadie, North-Holand Publishing Company, 1970.

REFERENCES FOR VOLUME II

4. R. New, "Optimal Planning Over Time — OPT," Journal of Systems Management, March 1972.

5. R. M. Soland, "An Algorithm For Separable Non-Convex Programming Problems II: Non-Convex Constraints," Management Science, Volume 17, No. 11, July 1971.

3 January 1972

RAC DISTRIBUTION LIST A

(Less these deletions: ______________________________)

Address code	Agency	Number of copies
	DEPARTMENT OF DEFENSE	
A2	Director of Defense Research and Engineering	1
A3	Assistant Secretary of Defense, (International Security Affairs)	1
A9	Assistant Secretary of Defense, (Systems Analysis)	2
B2	Joint Chiefs of Staff	1
B4	Studies Analysis Gaming Agency	1
B5	Weapons Systems Evaluation Group	1
C1	Advanced Research Projects Agency	1
C2	Defense Nuclear Agency	1
C3	National Security Agency	1
C4	Defense Communications Agency	1
* C6	Defense Documentation Center	12
C9	Defense Intelligence Agency	1
D1	National War College	1
D2	Industrial College of the Armed Forces	1
D3	Armed Forces Staff College	1
	HEADQUARTERS, DEPARTMENT OF THE ARMY	
E2	Office, Under Secretary of the Army (OR)	1
E4	Assistant Secretary of the Army (R&D)	1
E7	Deputy Chief of Staff for Personnel	1
E8	Deputy Chief of Staff for Military Operations	2
E9	Deputy Chief of Staff for Logistics	1
E10	Assistant Chief of Staff, Intelligence	2
E11	Directorate of Military Support	1
E14	Comptroller	1
** E15	The Army Library, Attn: ASDIRS	1
E16	Office of the Provost Marshal General	1
E17	Chief of Engineers	1
E18	Office of the Surgeon General	1
E19	Assistant Chief of Staff for Communications-Electronics	1
E22	Office of Personnel Operations	1
E23	Assistant Chief of Staff for Force Development	1
E27	Chief of Research and Development	1
E32	Office of Reserve Components	1
E33	Assistant Vice Chief of Staff	5
	ARMIES	
~~H2~~	~~First US Army~~	~~1~~
~~H3~~	~~Third US Army~~	~~1~~
~~H5~~	~~Fifth US Army~~	~~1~~
~~H6~~	~~Sixth US Army~~	~~1~~
~~H8~~	~~Eighth US Army~~	~~1~~
	UNIFIED COMMANDS	
~~L5~~	~~Commander in Chief, Alaska (CINCAL)~~	~~1~~
~~L6~~	~~Commander in Chief, Pacific (CINCPAC)~~	~~1~~
~~L12~~	~~Commander in Chief, Europe (CINCEUCOM)~~	~~1~~
~~L16~~	~~US Strike Command, MacDill Air Force Base (CINCSTRIKE)~~	~~1~~
	ARMY COMMANDS (CONUS)	
J11	US Army Security Agency	1
L1	US Army Air Defense Command	1
L61	US Continental Army Command	2
M19	US Army Strategic Communications Command	1
	ARMY COMMANDS (Overseas)	
~~L8~~	~~US Army, Alaska~~	~~1~~
~~L9~~	~~US Army Forces Southern Command~~	~~1~~
~~L10~~	~~US Army, Europe and Seventh Army~~	~~1~~

* Required IAW AR 70-11 and AR 70-31.

** Normally required; exclusion must be justified by Sponsor (AR 1-28).

RAC Distribution List A (continued)

3 January 1972

Address code	Agency	Number of copies
~~L11~~	~~US Army Pacific~~	~~1~~
~~R11~~	~~US Army Concept Team, Vietnam~~	~~1~~
	ARMY MATERIEL COMMAND	
L3	Hq US Army Materiel Command, Washington, D.C.	15
I5	US Army Munitions Command, Edgewood Arsenal	1
J4	US Army Armor and Engineer Board, Ft. Knox	1
J5	US Army Field Artillery Board, Ft. Sill	1
J6	US Army Aviation Test Board, Ft. Rucker	1
J8	US Army Infantry Board, Ft. Benning	1
M7	US Army Electronics Command, Ft. Monmouth	1
M10	US Army Missile Command, Redstone Arsenal	1
M13	US Army Munitions Command, Dover, N.J.	1
M22	US Army Aviation Systems Command, St. Louis	1
M24	US Army Weapons Command, Rock Island	1
M25	US Army Mobility Equipment Command, St. Louis	1
M26	Management Information Systems Directorate	1
M29	USA Advanced Materiel Concepts Agency, Alexandria, Va.	1
M30	US Army Tank-Automotive Command, Warren	1
M32	US Army Test and Evaluations Command, Aberdeen Proving Ground	1
R6	Dugway Proving Ground, Dugway	1
R9	White Sands Missile Range, Los Cruces	1
G25	Director, Army Materiel Systems Analysis Agency, Aberdeen Proving Ground	1
	COMBAT DEVELOPMENTS COMMAND	
L2	Hq USA, Combat Developments Command, Ft. Belvoir	1
G2	USACDC Air Defense Agency, Ft. Bliss	1
G3	USACDC Armor Agency, Ft. Knox	1
G4	USACDC Field Artillery Agency, Ft. Sill	1
G5	USACDC Aviation Agency, Ft. Rucker	1
G7	USACDC Chemical, Biological, and Radiological Agency, Ft. McClellan	1
G11	USACDC Communications-Electronics Agency, Ft. Monmouth	1
G12	USACDC Engineer Agency, Ft. Belvoir	1
G13	USACDC Infantry Agency, Ft. Benning	1
G16	USACDC Military Police Agency, Ft. Gordon	1
G19	USACDC Transportation Agency, Ft. Eustis	1
G20	USACDC Intelligence Agency, Ft. Huachuca	1
~~G22~~	~~USACDC Nuclear Agency, Ft. Bliss~~	~~1~~
~~G26~~	~~USACDC Special Operations Agency, Ft. Bragg~~	~~1~~
G28	USACDC Maintenance Agency, Aberdeen Proving Ground	1
L44	USACDC Concepts and Force Design Group, Alexandria	1
L45	USACDC Systems Analysis Group, Ft. Belvoir	1
L69	USACDC Intelligence and Control Systems Group, Ft. Belvoir	1
O26	USACDC Strategic Studies Institute, Carlisle Barracks	1
P2	USACDC Combat Systems Group, Ft. Leavenworth	1
P3	USACDC Personnel and Logistics Systems Group, Ft. Lee	1
	SCHOOLS, US ARMY	
~~G27~~	~~US Army Military Police, Ft. Gordon~~	~~1~~
~~M17~~	~~US Army Ordnance, Aberdeen Proving Ground~~	~~1~~
~~O2~~	~~US Army Air Defense, Ft. Bliss~~	~~1~~
~~O3~~	~~US Army Armor, Ft. Knox~~	~~1~~
~~O4~~	~~US Army Field Artillery, Ft. Sill~~	~~1~~
~~O5~~	~~US Army Chemical, Ft. McClellan~~	~~1~~
~~O6~~	~~US Army Engineer, Ft. Belvoir~~	~~1~~
~~O7~~	~~US Army Finance, Ft. Benjamin Harrison~~	~~1~~
~~O8~~	~~US Army Infantry, Ft. Benning~~	~~2~~
~~O9~~	~~US Army Intelligence, Ft. Huachuca~~	~~1~~
~~O10~~	~~Medical Field Service, Brooke Army Medical Center~~	~~1~~
~~O11~~	~~US Army Command and General Staff College, Ft. Leavenworth~~	~~1~~
~~O12~~	~~US Army Aviation, Ft. Rucker~~	~~1~~
~~O16~~	~~US Army School, Europe~~	~~1~~
~~O19~~	~~US Army Missile and Munitions Center and School, Redstone Arsenal~~	~~1~~

RAC Distribution List A (continued)

3 January 1972

Address code	Agency	Number of copies
~~O21~~	~~US Army Quartermaster, Ft. Lee~~	~~1~~
~~O23~~	~~US Army Institute of Military Assistance, Ft. Bragg~~	~~2~~
~~O24~~	~~US Army War College, Carlisle~~	~~1~~
~~O25~~	~~US Army Transportation, Ft. Eustis~~	~~1~~
~~O27~~	~~USMA (Academic Computer Center), West Point~~	~~1~~
O29	US Army Adjutant General, Ft. Benjamin Harrison	1
O30	US Army Combat Surveillance School and Training School, Ft. Huachuca	1
O31	US Army Signal Center and School, Ft. Monmouth	1
O32	US Army Southeastern Signal School, Ft. Gordon	1
O33	USWAC School, Ft. McClellan	1
	MISCELLANEOUS ARMY (CONUS)	
E34	US Army Intelligence Threat Analysis Detachment	1
E37	Logistics Doctrine, Systems and Readiness Agency, New Cumberland Army Depot	1
E38	Engineer Strategic Studies Group	1
K11	US Army Logistics Management Center	2
P9	US Army Strategy and Tactics Analysis Group	1
R4	US Army Behavioral and Systems Research Laboratory	1
	US AIR FORCE	
T4	Hq, US Air Force (AF/SAMID)	1
T8	Air University Library, Maxwell Field	1
	US NAVY	
S1	Chief of Naval Operations, OP-96	1
S2	Chief of Naval Operations, OP03EG-CNO	1
S9	Naval War College, Newport	1
	US MARINE CORPS	
S23	Marine Corps Development and Education Command, Quantico, Va.	1
	US GOVERNMENT AGENCIES	
~~U1~~	~~Central Intelligence Agency~~	~~1~~
~~U6~~	~~Department of State, Foreign Affairs Research Documents Center~~	~~3~~
U9	Department of State, Office of Science and Technology	1
	DEFENSE CONTRACTORS AND UNIVERSITIES	
C216	Center for Naval Analysis	1
V2	Human Resources Research Organization, Inc.	1
V3	Institute for Defense Analyses	1
V5	RAND Corporation	1
V6	Center for Research in Social Systems	1
V7	Stanford Research Institute	1
	FOREIGN GOVERNMENTS—BASIC STANDARDIZATION AGREEMENT COUNTRIES (Released through ST&A Division, OCRD)	
L7	Supreme Headquarters, Allied Powers, Europe (USNMR)*	1
W1	British Defense Research Staff	2
W2	Canadian Defense Research Staff	2
W3	North American Air Defense Command (US-Canadian HQ)	1
W4	US Army Strategic Group–UK (for release to DOAE)	1
W5	Australian Army Representative	2
W6	US Delegation, UN Military Staff Committee	1

*Foreign Headquarters US Representative.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
RESEARCH ANALYSIS CORPORATION

2a. REPORT SECURITY CLASSIFICATION
none

2b. GROUP
N/A

3. REPORT TITLE
A Methodology for Optimal Planning Over Time, Volume II, Appendices A,B,C,D,&E

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Paper

5. AUTHOR(S) *(First name, middle initial, last name)*
Charles A. Allen
Beverly D. Causey
James E. Falk
Ronald G. Magee
Ronald New, Project Director
John D. Pearson
Philip D. Robers
Charles W. Mylander

6. REPORT DATE
January 1972

7a. TOTAL NO. OF PAGES
284

7b. NO. OF REFS
2

8a. CONTRACT OR GRANT NO.

b. PROJECT NO. 011.310

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
TP-445, Final Draft of Vol II

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
Approved for public release;
distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
US Army, Combat Developments Command
Combat Systems Group

13. ABSTRACT

This report describes a methodology which can be used to identify the most cost-effective plan for the phase-in and phase-out of vehicle systems--a methodology for optimal fleet planning over time. Volume I provides a systematic development of the problem structure, a qualitative description of the solution procedure, and mathematical and operational descriptions of the algorithm. Volume II provides appendices containing a demonstration problem, subroutine descriptions, program flow charts, program listings, and error message descriptions.

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
branch and bound						
fleet planning						
non-convex programming						
non-linear programming						
optimization over time						
vehicle systems						

UNCLASSIFIED
Security Classification